

NOTE

Les éléments qui suivent, listings et documentations sont produits pour la satisfaction et l'usage personnel par la Société de Micro-informatique et Télécommunications SMT.

Toute reproduction, même partielle, est interdite et constitue une contrefaçon passible des peines prévues par la loi du 11 mars 1975 sur la protection des droits d'auteur.

L'application de cette règle permettra de vendre de plus en plus de programmes à des prix de plus en plus bas.

Son respect est la condition nécessaire pour que la SMT, ou d'autres sociétés, puissent produire, acheter ou adapter toujours plus de logiciels de qualité et que la créativité en ces matières se développe à votre profit.

PREFACE

Le propos de ce manuel est de fournir à l'utilisateur toutes les informations nécessaires à l'utilisation des sous programmes et des fonctions offertes par le système d'exploitation FLEX. Ce manuel peut être utilisé pour toutes les versions du FLEX (versions 5", 8"...). Il est déconseillé de l'ouvrir avant d'être parfaitement familiarisé avec l'utilisation du système GOUPIL et de ses disques.

Ce manuel est destiné aux programmeurs ayant acquis une bonne expérience des techniques de programmation en langage assembleur.

Il s'applique à la dernière version G2FLEX de l'adaptation du système FLEX au micro-ordinateur GOUPIL.

SOMMAIRE

	<u>Page</u>
Avertissement	1
1. Introduction	2
2. Le moniteur d'exploitation disque	3
2.1 - Présentation	3
2.2 - Variables globales	3
2.3 - Carte mémoire des variables et paramètres	3
2.4 - Sous programmes du moniteur	4
3. Ecriture d'utilitaires par l'utilisateur	19
3.1 - Commandes résidentes en mémoire	19
3.2 - Commandes résidentes sur disque	20
3.3 - Remarques d'ordre général	22
3.4 - Exemple d'utilisation	23
4. Le système de gestion des fichiers (FMS)	24
4.1 - Le bloc de contrôle du fichier (FCB)	26
4.1bis Points d'entrée du système de gestion de fichier	31
4.2 - Variables globales du FMS	32
4.3 - Codes fonction du FMS	33
5. Fichiers à accès direct	44
6. Erreurs FLEX	46
7. Informations particulières sur les disques et les fichiers	50
7.1 - Modules d'interface du disque	50
7.2 - Initialisation des disquettes	52
7.3 - Description d'un secteur de l'inventaire	54
7.4 - Description d'un fichier binaire	55
7.5 - Description d'un fichier source	56
7.6 - Création de commandes utilitaires	57
7.7 - Commande utilitaire : LINK	61
7.8 - Modules d'impression PRINT.SYS et P	62
7.9 - Les interruptions dans FLEX	66

AVERTISSEMENT

Le logiciel doit être utilisé selon les indications données dans ce document et le manuel d'utilisation. La SMT ne sera pas responsable d'un usage incorrect et destructif des fonctions et des paramètres, du système d'exploitation. Le lecteur est invité à porter son attention sur les diverses mises en garde qu'il trouvera dans ce document.

En aucun cas, la SMT ne sera tenue pour responsable de destructions provoquées par des modifications du produit effectuées sans son accord.

1 - INTRODUCTION

FLEX peut être divisé en trois grandes parties qui sont :

- le DØS ("Disk Operating System") qui exécute les commandes,
- le FMS ("File Management System") qui réalise la gestion des fichiers sur les disquettes,
- l'UCS ("Utility Command Set") qui regroupe l'ensemble des commandes utilitaires dont dispose l'utilisateur.

Les commandes sont décrites dans le manuel d'utilisation du FLEX. Nous nous intéresserons ici aux fonctions du DOS et du FMS afin de permettre au programmeur d'écrire ses propres commandes et de mettre en oeuvre une application utilisant des fichiers disques.

Lors de la mise au point des programmes qui traitent des fichiers sur disque avec l'aide de FMS, le programmeur doit prendre les précautions suivantes :

- (i) protéger en écriture la disquette système en couvrant ou découvrant la découpe de protection de la disquette (voir le manuel d'utilisation pour plus de détails sur cette opération). Ceci afin d'éviter une éventuelle destruction du système en cas d'erreur de programmation, l'exécution d'un programme erroné pouvant avoir des effets inattendus et dévastateurs.
- (ii) utiliser une disquette de travail pour la création ou le traitement de fichiers par un programme. Si des données doivent être préservées, en établir une copie pour ne pas les perdre en cas de fausse manoeuvre et de destruction du contenu de la disquette.
- (iii) tester le programme dans les "cas anormaux" où une donnée lue peut ne pas correspondre à ce que le programme attend. Un programme bien écrit doit pouvoir détecter les erreurs et contrôler sa bonne exécution: C'est à dire que lorsqu'une erreur se produit au cours d'un traitement, ou d'une opération d'entrée/sortie par exemple, il faut prévoir un traitement particulier de reprise ou carrément d'abandon du programme.

2 - LE MONITEUR D'EXPLOITATION DISQUE

2.1. PRESENTATION

Le DØS, système d'exploitation disque, assure le lien entre l'utilisateur, via le clavier et l'écran, et le système de gestion des fichiers disque. Les commandes sont lues et traitées par le DOS. Les fonctions telles que l'analyse syntaxique des noms de fichiers (nom + extension + numéro d'unité disque), ou des arguments des commandes, la gestion des entrées/sorties du clavier et de l'écran, les rapports d'erreur sont aussi toutes assurés par le DØS.

Les paragraphes suivants donnent la définition des variables globales du DØS et leur adresse en mémoire, et décrivent les sous programmes appelables par l'utilisateur.

2.2. VARIABLES GLOBALES

Toutes les variables intéressant le programmeur sont rangées dans deux zones mémoires du DØS, \$C080 - \$COFF et \$CC00 - CCFF. Certains emplacements de ces zones sont réservés à l'usage de celui-ci (constante, mémoire de travail,...) et ne doivent en aucun cas être utilisés par un programme de l'utilisateur qui risquerait de provoquer des destructions.

2.3. CARTE MEMOIRE DES VARIABLES ET PARAMETRES

2.3.1. TAMPON LIGNE

\$C080 - \$COFF : tampon ligne.

Bloc de 128 octets où le sous programme INBUF range les caractères frappés au clavier. Tous les caractères, à l'exception des caractères de contrôle sont placés dans le tampon. Les caractères effacés par la touche "retour arrière" ne sont pas mémorisés dans le tampon, ainsi que la touche "retour arrière" elle-même. Le retour chariot signalant la fin de l'entrée d'une ligne est rangé dans le tampon. Le tampon est aussi utilisé pour charger le fichier STARTUP lors d'un démarrage à froid du système disque (commande ).

Manuel de Programmation Avancée

2.3.2. VARIABLES DE CONTROLE DE L'ENVIRONNEMENT DU SYSTEME

\$CC00 : caractère de retour arrière TTYSET BS

C'est le caractère interprété par le sous programme INBUF comme retour arrière (backspace). Il peut être défini par l'utilisateur au moyen de la commande utilitaire TTYSET. Sa valeur par défaut est le caractère ASCII BS, soit \$08 en hexadécimal (ou aussi CTRL - H).

\$CC01 : caractère d'effacement TTYSET DL

C'est le caractère interprété par le sous programme INBUF comme caractère d'effacement ou d'annulation de ligne. Il peut être défini par l'utilisateur au moyen de la commande TTYSET. C'est par défaut le caractère ASCII CAN (ou CTRL - X) soit \$18.

\$CC02 : caractère séparateur (EOL) TTYSET EL

C'est le caractère séparant plusieurs commandes dans une ligne (commande multiple). Il peut être défini par l'utilisateur au moyen de la commande TTYSET. Valeur par défaut \$3A, caractère ASCII deux points ":".

\$CC03 : nombre de lignes par page TTYSET DP

Cet octet détermine le nombre de lignes que FLEX imprime sur une page avant de se mettre en pause ou de commander un saut à la page suivante. Il peut être défini par l'utilisateur au moyen de la commande TTYSET. Sa valeur par défaut est 24.

\$CC04 : nombre de caractères par ligne TTYSET WD

Cet octet fixe le nombre de caractères par ligne. S'il est nul, il n'y a pas de limite à la longueur de la ligne. Sa valeur est 80 par défaut et peut être redéfinie au moyen de la commande TTYSET.

\$CC05 : temporisation retour chariot

Cet octet donne le nombre de caractères nuls ou de remplissage à envoyer après la sortie d'un retour chariot de façon à tenir compte du temps de retour chariot non négligeable de certains terminaux imprimants. Sa valeur par défaut est 0 et peut être redéfinie par la commande TTYSET.

\$CC06 : caractère de tabulation TTYSET TB

Cet octet définit un caractère de tabulation qui peut être utilisé par des programmes tel que l'éditeur. Le FLEX ne l'utilise pas. Sa valeur par défaut est 0, il n'est pas défini et peut être fixé par la commande TTYSET.

\$CC07 : caractère écho du retour arrière TTYSET BE

C'est le caractère que le sous programme INBUF envoie comme écho à la réception du caractère retour arrière. Si il a la même valeur \$08 que le caractère retour arrière, FLEX envoie un espace \$20 avant l'écho ce qui permet d'effacer sur un écran le caractère affecté par le retour arrière. Sa valeur par défaut est 08.

\$CC08 : nombre de lignes sautées entre deux pages TTYSET EJ

Cet octet indique à FLEX le nombre de lignes à sauter entre deux pages, une page ayant le nombre de lignes défini en \$CC03. La valeur par défaut est 0, il n'y a pas de mise en page, elle peut être redéfinie par la commande TTYSET.

\$CC09 : drapeau de pause entre pages TTYSET PS

Ce drapeau permet d'arrêter la sortie entre chaque page. Si l'octet est nul, le FLEX arrête la sortie (sur écran ou imprimante) à la fin de chaque page, s'il est non nul il n'y a pas de pause. Le caractère d'échappement permet de reprendre l'édition. Sa valeur par défaut est 0 et peut être modifiée par la commande TTYSET.

\$CC0A : caractère d'échappement TTYSET ES

La frappe de ce caractère provoque l'arrêt de l'édition en fin de ligne. Sa valeur par défaut est \$20, le caractère ASCII espace. Pour reprendre la sortie, frapper à nouveau ce caractère.

2.3.3. VARIABLES SYSTEMES

\$CC0B : numéro de l'unité système

C'est le numéro de l'unité de disquette à partir de laquelle les commandes sont chargées. Si la valeur de cet octet est \$FF, la recherche se fera sur toutes les unités prêtes. L'unité par défaut est l'unité 0. Peut être assigné par la commande ASN.

Manuel de Programmation Avancée

\$CCOC : numéro de l'unité de travail

C'est le numéro choisi par défaut de l'unité de disquette utilisé pour la recherche ou la création d'un fichier. Si la valeur de cet octet est \$FF, la recherche se fera sur toutes les unités prêtes. Sa valeur par défaut est 1. Peut être assigné par la commande ASN.

\$CCOD : réservée au système

\$CCOE - \$CCIO : date

Ces trois octets sont utilisés pour ranger la date donnée lors du chargement du moniteur. Elle est mémorisée en binaire, le jour dans le second octet, le mois dans le premier et l'année dans le troisième (chiffre des unités et dizaines seulement).

\$CC11 : caractère terminal

Cet octet contient le dernier caractère non alphanumérique rencontré dans la lecture séquentielle du tampon de ligne. Voir les commentaires sur les sous programmes NXTCH et CLASS dans le paragraphe sous programmes système appelables par l'utilisateur.

\$CC12 - \$CC13 : adresse de la table de commande utilisateur.

Le programmeur peut ranger dans ces deux octets l'adresse de sa table de commande. Se référer au chapitre concernant les commandes écrites par l'utilisateur pour les détails. La valeur de cette adresse est 0 par défaut, c'est à dire qu'il n'y a pas de table de commande utilisateur.

\$CC14 - \$CC15 : pointeur du tampon de ligne

Ces deux octets contiennent l'adresse du prochain caractère à traiter dans le tampon de ligne. Voir les sous programmes INBUFF, NXTCH, GETFIL, GETCHR et DØCMND dans le paragraphe traitant des sous programmes système.

\$CC16 - \$CC17 : adresse de retour sur échappement

Ces deux octets contiennent l'adresse de retour si la frappe d'un retour chariot suit celle d'un caractère d'échappement pour arrêter une sortie. Se référer à la description de la commande TTYSET dans le manuel d'utilisation du FLEX pour plus d'informations sur la procédure d'échappement. Voir aussi la documentation du sous programme PCRLF.

\$CC18 : caractère courant

Cet octet contient le dernier caractère traité par le sous programme NXTCH dans le tampon ligne. Voir le sous programme pour plus de détails.

\$CC19 : caractère précédent

Cet octet contient le précédent caractère traité par le sous programme NXTCH dans le tampon ligne. Voir le sous programme pour plus de détails.

\$CC1A : numéro de la ligne courante

Cet octet contient le nombre de lignes sorties dans une page. Comparé au nombre de lignes par page, il permet de déterminer la fin de la page.

\$CC1B - \$CC1C : déplacement de l'adresse de chargement

Ces deux octets contiennent le déplacement à ajouter à l'adresse de chargement d'un programme à partir du disque. Voir le sous programme LØAD pour plus de détails. Cet emplacement est aussi utilisé comme mémoire de travail par certains modules du FLEX.

\$CC1D : drapeau de transfert

Après le chargement en mémoire d'un programme à partir du disque, cet octet est non nul si une adresse de transfert a été trouvée au cours du chargement. Il est aussi utilisé par certains modules du FLEX.

\$CC1E - \$CC1F : adresse de transfert

Si le drapeau de transfert est non nul après le chargement d'un programme, ces deux octets contiennent la dernière adresse de transfert rencontrée (c'est à dire l'adresse d'exécution du programme). Si le drapeau est à zéro, le contenu est indéterminé.

\$CC20 : code d'erreur

Cet octet contient le numéro d'erreur retourné en cas d'erreur par plusieurs des fonctions du système de gestion des fichiers (FMS).

8.

Manuel de Programmation Avancée

\$CC21 : drapeau d'entrée/sortie spéciale

Si cet octet est non nul, le sous-programme PØTCHR ignore le nombre de caractères par ligne (TTYSET WD) et le caractère d'échappement (TTYSET ES). Le sous-programme RSTRIØ remet le drapeau à 0. La valeur par défaut est 0.

\$CC22 : aiguillage de la sortie

Si cet octet est nul, le sous-programme de sortie PUICHR utilise le sous programme ØUTCH, sinon le sous-programme ØUTCH2. Voir la description de ces sous-programmes pour plus de détails.

\$CC23 : aiguillage de l'entrée

Si cet octet est nul, le sous-programme d'entrée GETCHR utilise le sous programme INCH, sinon le sous-programme INCH2. Voir la description de ces sous-programmes pour plus de détails.

\$CC24 - \$CC25 : adresse du FCB de sortie

S'ils ne sont pas nuls, ces deux octets contiennent l'adresse d'un FCB (bloc de contrôle de fichier) utilisé pour la sortie dans un fichier. S'ils sont nuls, la sortie n'est pas exécutée. Voir le sous programme ØUTCHR pour plus de détails. Les deux octets sont remis à 0 par le sous programme RSTRIØ.

\$CC26 - \$CC27 : adresse du FCB d'entrée

S'ils ne sont pas nuls, ces deux octets contiennent l'adresse d'un FCB (bloc de contrôle de fichier) utilisée pour l'entrée à partir d'un fichier. S'ils sont nuls, l'entrée n'est pas exécutée. Voir le sous programme GETCHR pour plus de détails. Le sous programme RSTRIØ remet à 0 ces deux octets.

\$CC28 : drapeau commande

Cet octet est non nul si le FLEX est appelé par un programme utilisateur via le point d'entrée DØCMND. Voir la documentation de DØCMND pour plus de détails.

\$CC29 : compte des caractères sortis

Cet octet contient le compte des caractères sortis dans une ligne. Il est comparé au nombre de caractères par ligne TTYSET WD pour déterminer la fin de la ligne. La sortie d'un caractère de contrôle remet ce compte à 0..

\$CC2A : réservé au système

\$CC2B - \$CC2C : adresse de fin de la mémoire utilisateur

Ces deux octets contiennent l'adresse de la dernière cellule mémoire de l'utilisateur. Le contenu est initialisé lors du chargement du système et peut être lu par le programme ayant besoin de cette information.

\$CC2D - \$CC2E : aiguillage fichier d'erreurs

Si ces deux octets sont nuls, le sous programme RPTERR utilise ERRORS.SYS comme fichier contenant les messages d'erreur. Sinon ils contiennent l'adresse du nom du fichier à utiliser. Voir le sous programme RPTERR pour plus de détails.

\$CC2F : drapeau d'écho de lecture fichier

Si cet octet est non nul (valeur par défaut) et que l'entrée est faite à partir d'un fichier, le caractère lu est envoyé sur le canal de sortie. Si l'octet est nul il n'y a pas d'écho.

\$CC30 - \$CC4D : réservés au système

\$CC4E - \$CCBF : constantes du système

\$CCCO - \$CCD7 : initialisation imprimante

Cet espace est réservé pour le chargement du sous programme d'initialisation de l'imprimante.

\$CCD8 - \$CCE3 : test de l'état de l'imprimante

Cet espace est réservé pour le chargement du sous programme de test de l'imprimante.

Manuel de Programmation Avancée

\$CCE4 - \$CCF7 : sortie imprimante

Cet espace est réservé pour le chargement du sous programme de sortie d'un caractère vers l'imprimante.

\$CCF8 - \$CCFF : réservés au système.

2.4. SOUS-PROGRAMMES DU MONITEUR

A défaut d'indications contraires, les sous programmes doivent être appelés par une instruction JSR et le contenu de tous les registres doit être présumé détruit par leur exécution.

2.4.1. POINTS D'ENTREE DU MONITEUR

\$CDOO COLD : point d'entrée à froid

Le chargeur disque se branche à cette adresse pour initialiser le système chargé en mémoire. Le DØS et le FMS sont initialisés, ensuite FLEX s'identifie par le message habituel et demande la date. Si un fichier STARTUP existe il est directement chargé et exécuté. Le point d'entrée est réservé au chargeur et ne doit pas être utilisé par le programmeur qui pourrait risquer de détruire le contenu de la disquette système.

\$CDO3 WARM : point d'entrée à chaud

Pour les programmes utilisateurs c'est le point d'entrée principal dans le DØS. Le branchement doit se faire par une instruction JMP. La pile système et le registre de retour sur échappement (\$CC16 - \$CC17) sont initialisés, et le moniteur est prêt à traiter de nouvelles commandes. Le moniteur vérifie le dernier caractère terminal (\$CC11), s'il trouve le caractère TTYSET EL (fin de ligne, séparateur de commandes) il traitera la commande en attente dans le tampon de ligne (commande multiple). Sinon il ira lire une nouvelle ligne au clavier. Si le DØS a été appelé par un programme utilisateur, via le point d'entrée DØCMND, le contrôle sera rendu au programme à la fin du traitement des commandes rangées dans le tampon de ligne.

\$CDO6 RENTER : point d'entrée directe

C'est un point d'entrée directe dans la bouche principale du DØS, aucune initialisation n'est effectuée. Le branchement se fait par une instruction JMP. Normalement ce point d'entrée est utilisé de manière interne par le système et les programmes utilisateurs n'en ont pas besoin.

2.4.2. ENTREE D'UN CARACTERE

\$CD09 INCH entrée caractère
\$CDOC INCH2 entrée caractère

Ces deux sous programmes lisent un caractère au clavier et le retournent au programme appelant dans le registre A. Ils peuvent être remplacés par des sous programmes adaptés à la configuration de l'utilisateur. Le sous programme GETCHR utilise normalement INCH, mais si le drapeau "d'aiguillage de l'entrée (\$CC23) est non nul, il utilisera INCH2. Un programme utilisateur peut changer le vecteur de branchement à l'adresse INCH pour utiliser un autre sous programme gérant par exemple, un lecteur de ruban papier. Le vecteur de branchement à l'adresse INCH2 ne doit jamais être modifié. Lors d'une entrée à chaud dans le DØS par le point WARMS, le drapeau d'aiguillage de l'entrée est remis à 0 et le vecteur de branchement INCH est réinitialisé pour pointer sur le même sous programme que le vecteur INCH2. Le sous programme RSTRIO effectue lui aussi cette réinitialisation. Ces sous programmes ne tiennent pas compte des paramètres TTY SET.

\$CD15 GETCHR : acquisition d'un caractère au clavier

Ce sous programme permet l'acquisition d'un caractère au clavier, et le retourne au programme appelant dans le registre A. Son appel provoque la remise à zéro du numéro de ligne courante (\$CC1A). Il prend en compte les paramètres TTY SET, son utilisation est donc préférable à celle du INCH. Si le drapeau aiguillage de l'entrée est non nul, le sous programme INCH2 est utilisé pour l'entrée. S'il est nul, les deux octets \$CC26 - \$CC27, adresse FCB d'entrée, sont testés. S'ils sont nuls, le caractère est lu par le sous programme INCH, sinon ils contiennent l'adresse d'un FCB pour lire le caractère dans un fichier précédemment ouvert. Les registres B et X sont sauvegardés.

2.4.3. SORTIE D'UN CARACTERE

\$CDOF ØUTCH sortie caractère
\$CD12 ØUTCH2 sortie caractère

A l'appel de ces sous programmes le registre A doit contenir le caractère à sortir vers le périphérique désiré, normalement l'écran. Les vecteurs ØUTCH et ØUTCH2 pointent de façon standard vers le même sous programme de sortie sur écran ; cependant ØUTCH peut être modifié par l'utilisateur pour exécuter un autre sous programme de sortie vers un périphérique différent, par exemple une imprimante.

Manuel de Programmation Avancée

Le sous-programme PUTCHR, décrit ci-dessous, utilise ØUTCH ou ØUTCH2 selon le contenu du drapeau "aiguillage de sortie" (§CC22). Lors d'une entrée à chaud dans le moniteur (WARMS) l'aiguillage de sortie est remis à zéro, et ØUTCH est réinitialisé pour pointer le même sous programme que ØUTCH2. Le sous programme RSTRIØ effectue la même réinitialisation. Ces sous programmes ne tiennent pas compte des paramètres TTY SET.

§CD18 PUTCHR : écriture caractère

Ce sous programme sort le caractère contenu dans le registre A vers le périphérique choisi, normalement l'écran, en prenant en compte les paramètres TTY SET. Si le drapeau d'E/S spéciale (§CC21) est nul, le compte des caractères est testé (§CC29) et si la ligne courante est pleine, on passe alors à la suivante. Si un caractère d'échappement TTY SET a été frappé au clavier, la sortie s'arrête à la fin de la ligne courante. Le sous programme ØUTCH2 est utilisé pour envoyer le caractère si l'aiguillage de sortie (§CC22) est non nul, sinon l'adresse FCB de sortie est testée (§CC24 - §CC25). Si elle n'est pas nulle, c'est l'adresse du FCB d'un fichier de sortie, précédemment ouvert, et le caractère est écrit dans le fichier. Si elle est nulle, le sous programme ØUTCH est utilisé pour envoyer le caractère, normalement vers l'écran, mais l'utilisateur peut modifier le vecteur ØUTCH pour exécuter un autre sous programme de sortie. Les registres B et X sont préservés.

2.4.4. LECTURE D'UNE LIGNE

§CD18 INBUFF : entrée dans le tampon ligne

Ce sous programme lit une ligne au clavier et la range dans le tampon ligne. Les caractères de retour arrière et d'effacement TTYSET BS et TTYSET DL sont traités. Tous les autres caractères de contrôle sont ignorés, à l'exception du retour chariot et du saut de ligne. Le retour chariot est placé dans le tampon à la fin de la ligne, le saut à la ligne crée un blanc dans le tampon et provoque un retour chariot sur l'écran pour continuer l'entrée sur une nouvelle ligne. 128 caractères peuvent être entrés dans le tampon y compris le retour chariot final. Au cas où plus de caractères seraient frappés, la ligne est tronquée au 127ième, un retour chariot étant placé dans le 128ième. A la sortie du sous programme, le pointeur du tampon de ligne (§CC14 - §CC15) est initialisé sur le premier caractère.

Attention : les commandes du système entrées au clavier sont gardées dans le tampon de ligne. L'utilisation de INBUFF par un programme utilisateur peut donc détruire des commandes qui n'ont pas encore été traitées par le moniteur. Il est donc prudent de réserver l'usage du tampon de ligne à l'entrée de commandes du moniteur, pour ne pas risquer des effets imprévisibles.

2.4.5. SORTIE D'UNE CHAÎNE DE CARACTÈRES

§CD1E PSTRNG : sortie d'une chaîne de caractère

Ce sous programme envoie un retour chariot et un saut de ligne avant de sortir la chaîne de caractères. A l'appel, le registre X doit contenir l'adresse du premier caractère de la chaîne, qui doit se terminer par le caractère ASCII EOT (§04). Les paramètres TTYSET sont pris en compte. Le registre B est préservé.

2.4.6. CLASSEMENT D'UN CARACTÈRE

§CD21 CLASS : classement d'un caractère

Le sous programme teste si un caractère est alphanumérique ou non. En entrée le caractère doit se trouver dans le registre A, en sortie, si le caractère est alphanumérique (une lettre ou un chiffre) le bit de retenue (carry) du code de condition est mis à zéro ; sinon la retenue est mise à un et le caractère rangé dans le "dernier caractère terminal" (§CC11). Tous les registres sont préservés.

2.4.7. TRAITEMENT DU RETOUR CHARIOT/SAUT DE LIGNE

§CD24 PCRLF : sortie retour chariot/saut de ligne

Ce sous programme envoie un retour chariot et un saut de ligne (caractères ASCII CR et LF) en prenant en compte un certain nombre de paramètres TTYSET. Avant toute chose, le sous programme teste si un caractère d'échappement (TTYSET ES) a été frappé au clavier pendant la sortie de la ligne précédente. Si oui, il attend l'entrée d'un autre caractère d'échappement pour continuer ou d'un retour chariot qui provoquera l'abandon du travail en cours. Sur un retour chariot, le sous programme met à zéro le dernier caractère terminal (§CC11) pour supprimer les éventuelles commandes restant dans le tampon de ligne et se branche à l'adresse contenue dans le registre de retour sur échappement. Si elle n'est pas modifiée par l'utilisateur, cette adresse est celle du point d'entrée à chaud du moniteur, WARMS. Sur un caractère d'échappement (si nécessaire), le

numéro de la ligne courante est testé : si la page est pleine et qu'une pause est demandée entre chaque page (TTYSET PS), le sous programme attend la frappe d'un autre caractère d'échappement ou d'un retour chariot comme ci-dessus. Toutes les pauses sont faites avant la sortie des deux caractères CR et LF, suivi du nombre de caractères nuls spécifiés par la temporisation retour chariot (TTYSET NL). Quand la page courante est pleine, le sous programme sort le nombre de saut de ligne spécifié en \$CC08 (TTYSET EJ). Le registre X est préservé.

2.4.8. LECTURE DU TAMPON LIGNE

\$CD27 NXTCH : lecture du caractère suivant

Ce sous programme permet de lire séquentiellement le tampon de ligne. Le caractère courant remplace le caractère précédent (\$CC19 : = (\$CC18), le caractère suivant désigné par le pointeur du tampon ligne devient le caractère courant (\$CC18). Les espaces multiples sont ignorés pour ne donner qu'un seul espace. Le pointeur du tampon ligne est augmenté de un pour désigner le caractère suivant, sauf si le caractère courant est un retour chariot ou un séparateur (TTYSET EL). Dans ce dernier cas, tout nouvel appel à NXTCH continuera à donner le retour chariot ou le séparateur, sans passer à la nouvelle commande éventuellement présente dans le tampon. NXTCH sort par le sous programme CLASS, c'est à dire que le caractère se trouve dans le registre A et que la retenue est à zéro s'il est alphanumérique. Les registres B et X sont préservés.

2.4.9. RESTAURATION DES VECTEURS D'E/S

\$CD2A RSTRIØ : restauration des vecteurs d'E/S

Ce sous programme restaure les valeurs initiales des vecteurs ØUTCH et INCH pour qu'ils pointent vers les mêmes sous programmes d'E/S que, respectivement, ØUTCH2 et INCH2. Les aiguillages d'entrée et de sortie (\$CC23 et \$CC22) sont remis à zéro, ainsi que les adresses des FCB d'entrée et de sortie. Les registres A et B sont préservés.

2.4.10. TRANSFERT DES SPECIFICATIONS D'UN FICHIER

\$CD2D GETFIL : lecture des spécifications d'un fichier

Ce sous programme est destiné au rangement dans un FCB (conférer le chapitre sur le gestionnaire de fichier) des spécifications d'un fichier (disque, nom, extension) se trouvant dans le tampon de la ligne. A l'entrée le registre X doit contenir

l'adresse du FCB, et le pointeur du tampon ligne doit désigner le premier caractère des spécifications du fichier. Si le disque n'est pas précisé, le numéro de l'unité de travail sera rangé dans le FCB. En sortie la retenue du code de condition est à zéro s'il n'y a pas d'erreur de format, à un sinon. Si l'extension n'est pas précisée, le programme appelant doit utiliser le sous programme SETEXT pour spécifier l'option par défaut. Le pointeur du tampon ligne est laissé sur le caractère suivant le séparateur, sauf si c'est un retour chariot ou une fin de ligne (:). Quand une erreur est détectée, le code d'erreur 21 est rangé dans le FCB. Le registre X est préservé.

2.4.11. CHARGEUR

\$CD30 LOAD : chargement d'un fichier

Ce sous programme est destiné au chargement en mémoire d'un fichier binaire résident sur disque. En entrée le FCB du système (en \$C840) doit contenir le nom du fichier préalablement ouvert en lecture binaire. Le fichier est chargé en mémoire à l'adresse normalement spécifiée dans le fichier lui-même, mais il est possible de le charger dans une zone mémoire différente en utilisant le déplacement donné en \$CC1B - \$CC1C. Le déplacement est ajouté à l'adresse de chargement lue dans le fichier ; attention au débordement de l'addition. Si une adresse de transfert est trouvée, elle n'est pas modifiée par le déplacement. Il faut noter aussi que le contenu du fichier n'est pas modifié par le déplacement et qu'il ne s'agit pas d'une procédure de relogement de programme. Si l'on ne veut pas de déplacement il faut le mettre à zéro avant l'appel du sous programme. En sortie, le drapeau de transfert (\$CC1D) est à zéro, si aucune adresse de transfert n'a été trouvée, sinon il est non nul et la dernière adresse trouvée est rangée en \$CC1E - \$CC1F. Le fichier est fermé par le sous programme et si une erreur se produit, un message d'erreur est sorti et le contrôle est transféré au FLEX, au point d'entrée à chaud WARMS.

2.4.12. RANGEMENT DE L'EXTENSION DANS UN FCB

\$CD33 SETEXT : Définition de l'extension

Le sous programme permet de définir l'extension d'un fichier si elle n'est pas déjà précisée dans le FCB. En entrée, le registre X doit contenir l'adresse du FCB, le registre A le code numérique de l'extension :

0	BIN	Binaire
1	TXT	Texte
2	CMD	Commande
3	BAS	Basic source
4	SYS	Système
5	BAK	Backup éditeur
6	SCR	

16.

Manuel de Programmation Avancée

7	DAT	Données
8	BAC	Basic compilé
9	DIR	
10	PRT	
11	ØUT	Fichier imprimante (Spooler).

Si le FCB contient déjà une extension, à la suite d'un appel à GETFIL par exemple, le sous programme ne fait rien. Tout code différent de ceux donnés ci-dessus est ignoré. Le registre X est préservé.

2.4.13. ADDITION DU REGISTRE B AU REGISTRE X

§CD36 ADDBX : addition de B à X

Ce sous programme ajoute le contenu du registre B au contenu du registre X. Le contenu de B est détruit à la sortie.

2.4.14. SORTIE D'UN NOMBRE DECIMAL

§CD39 ØUTDEC : sortie d'un décimal

En entrée, le registre X doit contenir l'adresse de l'octet de poids fort d'un nombre binaire non signé sur 16 bits. Le registre B doit contenir un drapeau de suppression d'espace, s'il est non nul les zéros de tête sont remplacés par des blancs, s'il est nul l'impression commence avec le premier chiffre non nul.

2.4.15. SORTIE D'UN NOMBRE HEXADECIMAL

§CD3C ØUTHEX : sortie d'un hexadécimal

En entrée le registre X doit contenir l'adresse d'un octet qui sera imprimé sous la forme de deux chiffres hexadécimaux. Les registres B et X sont préservés.

2.4.16. RAPPORT D'ERREUR

§CD3F RPTERR : message d'erreur

Ce sous programme assure la sortie d'un message en cas de retour en erreur d'une fonction du gestionnaire de fichier FMS. Les messages d'erreur système sont stockés dans le fichier disque ERRØRS.SYS. mais le programmeur peut utiliser son propre fichier de messages. En entrée, le registre X doit contenir l'adresse d'un FCB dont le code d'erreur est non nul. Ce code est transféré en §CC20, un appel au sous programme RSTRIØ est effectué et l'aiguillage fichier d'erreurs est testé (§CC2D - §CC2E). Si l'aiguillage est nul, le fichier ERRØR.SYS est ouvert en accès direct pour la recherche du message, sinon c'est l'adresse d'une chaîne ASCII donnant les spécifications du fichier à utiliser (non + extension sur 11 caractères). Le fichier utilisateur est alors ouvert en accès direct. Le code d'erreur est utilisé pour calculer le numéro de l'enregistrement dans lequel se trouve le message et son emplacement dans l'enregistrement. Sachant que chaque message

est rangé dans une zone de 63 caractères, à raison de quatre zones par enregistrement (un enregistrement correspondant à un secteur disque). Si le message est trouvé il est sorti sur l'écran, sinon (le code d'erreur n'étant pas répertorié ou le fichier n'ayant pas été trouvé) le code d'erreur nnn est affiché sur l'écran avec le message :

ERR.DISQUE # nnn

La description des codes d'erreur système est donnée dans la suite du manuel.

2.4.17. CONVERSION D'UN NOMBRE HEXADÉCIMAL

\$CD42 GETHEX : entrée d'un nombre hexadécimal

Ce sous programme permet de convertir en binaire un nombre hexadécimal entré dans le tampon de ligne. En entrée, le pointeur tampon de ligne doit désigner le premier caractère du nombre ; en sortie la retenue est à zéro si le sous programme a trouvé un nombre valide, le registre B est non nul, et le registre X contient le nombre en binaire. Le pointeur tampon de ligne est laissé sur le premier caractère suivant le séparateur sauf si ce séparateur est un retour chariot ou une fin de ligne.

La valeur retournée dans le registre X est nulle si le premier caractère examiné dans le tampon est un séparateur (tel qu'une virgule), la retenue est mise à zéro ainsi que le registre B pour indiquer que le nombre n'a pas été trouvé (ou bien qu'il a été omis intentionnellement, sachant qu'il prendra la valeur 0 par défaut). Si un caractère non hexadécimal est trouvé, les suivants sont sautés jusqu'au prochain séparateur, et la retenue est mise à 1. La longueur du nombre dans le tampon n'est pas limitée mais sa valeur binaire est tronquée à \$FFFF incluse.

2.4.18. SORTIE D'UNE ADRESSE EN HEXADÉCIMAL

\$CD45 ØUTADR : sortie d'une adresse hexadécimale

En entrée, le registre X contient l'adresse de l'octet de poids fort d'une adresse binaire codée sur 2 octets. Cette valeur est sortie codée en hexadécimal avec quatre chiffres.

2.4.19. CONVERSION D'UN NOMBRE DÉCIMAL

\$CD48 INDEC : entrée d'un nombre décimal

Ce sous programme permet de convertir en binaire un nombre décimal non signé entré dans le tampon ligne.

En entrée, le pointeur du tampon de ligne doit désigner le premier caractère du nombre. En sortie, la retenue est mise à zéro si un nombre valide a été trouvé, le registre B est non nul et le registre X contient la valeur binaire de ce nombre. Le pointeur du tampon est laissé sur le premier caractère suivant le séparateur, sauf si ce séparateur est un retour chariot ou une fin de ligne. Si le premier caractère examiné dans le tampon est un séparateur (une virgule, par exemple), la retenue est mise à zéro, ainsi que le registre B pour indiquer l'absence du nombre, et la valeur retournée dans X est nulle. La longueur du nombre dans le tampon n'est pas limitée mais le résultat est tronqué à la précision des 16 bits.

2.4.20. UTILISATION DU DØS COMME SOUS PROGRAMME

\$CD4B DØCMND : appel du DØS

Ce point d'entrée dans le moniteur permet à un programme de faire exécuter une commande par le moniteur et de récupérer le contrôle à la fin de la commande (qui peut être multiple). La commande doit être placée dans le tampon de ligne par le programme, et le pointeur doit être initialisé sur le premier caractère de la commande. Il faut noter que cette opération détruit le contenu du tampon et que l'on doit veiller à ne pas rompre l'enchaînement de commandes multiples. La commande doit être terminée par un retour chariot (\$D). Le DØS rend le contrôle au programme utilisateur après l'exécution de la commande avec le registre B à zéro si aucune erreur n'a été détectée, avec le code d'erreur dans B en cas d'erreur (du FMS s'il est utilisé).

Attention ! ne pas utiliser cette procédure pour charger un programme qui pourrait recouvrir le programme appelant en mémoire et le détruire.

Un exemple d'utilisation pourrait être la sauvegarde sur disque d'une partie de la mémoire dans un fichier binaire, en construisant une commande SAVE dans le tampon avec le nom du fichier et les paramètres et en appelant le moniteur par ce point d'entrée.

3 - ECRITURE D'UTILITAIRES PAR L'UTILISATEUR
--

La grande souplesse de FLEX, est que l'utilisateur peut écrire ses propres commandes utilitaires. Elles peuvent être stockées sur disque ou résidentes en mémoire vive (RAM) ou morte (ROM-EPRM).

3.1 - Commandes résidentes en mémoire

De telles commandes sont des programmes, toujours résidents en mémoire, auxquels le DOS fait appel lorsque le nom associé à la commande est entré au clavier.

Les noms des commandes et les points d'entrée (adresse d'exécution) des programmes associés sont stockés dans une table créée par l'utilisateur en mémoire. Le format de cette table, est défini ci-après :

```
FCC "nom de la commande"
FCB 0
FDB "adresse d'exécution de la commande".
```

La table des commandes utilisateur doit toujours se terminer par un octet nul.

Par exemple :

```
FCC "DEBUG"           1ère commande utilisateur
FCB 0
FDB $3000             adresse d'exécution du pgm DEBUG
FCC "PUNT"            2ème commande utilisateur
FCB 0
FDB $3200             adresse d'exécution du pgm PUNT
FCB 0                 fin de la table.
```

L'adresse de la table des commandes utilisateur doit être fournie au DØS en \$CC12-\$CC13 (conférer carte mémoire).

Lorsqu'une commande est entrée au clavier, le DØS explore successivement sa propre table des commandes, la table des commandes utilisateur (si elle existe) puis le catalogue du disque.

La table du DØS contient uniquement les commandes GET et MØN. Comme elle est explorée avant celle de l'utilisateur, celui-ci ne pourra pas créer des commandes répondant aux noms GET et MØN. Par contre, il pourra créer des commandes aux noms identiques à ceux des commandes répertoriées dans le catalogue du disque, tout en sachant que ces dernières ne pourront plus être exécutées tant que l'accès à la table des Commandes Utilisateur n'est pas inhibé, en remettant à zéro le pointeur \$CC12-\$CC13.

3.2 - Commandes résidentes sur disque

Les commandes utilisateurs sauvegardées sur disque doivent obligatoirement posséder une adresse de transfert et une extension CMD.

En général, ces commandes sont créées en assembleur, et la manière d'affecter une adresse de transfert à un programme en cours d'assemblage, est décrite dans le manuel de présentation de l'assembleur (ASMB).

Une commande utilisateur peut être exécutée en n'importe quel endroit de la mémoire RAM utilisateur. Son adresse de chargement est précisée à l'assemblage par la directive ØRG.

Mais dans la plupart des cas, il est utilisé de pouvoir les charger dans l'espace des commandes réservé par le FLEX (\$C100-\$C6FF).

Pratiquement toutes les commandes fournies avec FLEX s'exécutent dans cette zone mémoire. La commande SAVE ne fait pas exception à la règle, aussi il n'est pas possible de l'utiliser pour sauvegarder des informations contenues dans cette zone. C'est pour cette raison que la commande SAVE.LØW a été créée. Elle est identique à la commande SAVE mais s'exécute en \$0100.

La table FCB est utilisée pour le chargement des commandes disque, les commandes écrites pour s'exécuter dans l'espace des commandes utilitaires ne doivent donc pas empiéter dessus.

Par contre, une fois chargées, elles peuvent l'utiliser comme zone tampon ou FCB pour leurs propres accès disque.

Pour plus de détails se référer aux exemples donnés dans le chapitre descriptif du FMS.

3.3 - Remarques d'ordre général

Les commandes écrites par l'utilisateur sont appelées par l'instruction de branchement JMP. En retour, elles doivent redonner la main au DØS en se branchant (instruction JMP) au point d'entrée à chaud du FLEX \$CD03.

3.3.1. - Acquisition des arguments

Les commandes écrites par l'utilisateur peuvent être amenées à utiliser des arguments entrés au clavier. Le nom de la commande et les arguments sont stockés dans le buffer de ligne (\$0080 - \$00FF). A l'entrée de la commande, le pointeur du buffer de ligne adresse la position du 1er caractère du 1er argument s'il existe, sinon il désigne soit la position du caractère de fin de ligne de commande (:), soit la position du retour chariot (↵).

Les sous-programmes NXTCH, GETFIL et GETHEX peuvent être utilisés pour faire l'acquisition de ces arguments.

3.3.2. - Traitement des erreurs

Si en cours d'exécution une erreur est détectée par le DØS, le FMS ou par la commande elle-même, il faut prévoir un traitement de reprise ou l'abandon de l'exécution de la commande en redonnant le contrôle au FLEX par le point d'entrée à chaud (\$CD03). Après avoir affiché un message approprié (sous programme RPTERR par exemple pour une erreur FMS).

3.4 - Exemple d'utilisation

1 - L'écriture du nom d'un fichier dans le FCB peut être réalisée de la manière suivante :

On suppose que le pointeur du buffer de ligne désigne le 1er caractère du nom du fichier et que son extension sera TXT par défaut.

LDX	# FCB	(X) pointe sur le FCB
JSR	GETFIL	écriture du nom du fichier dans le FCB
BCS	ERRØR	traitement d'erreur
LDAA	# 1	code de l'extension TXT
JSR	SETEXT	écriture de l'extension par défaut

Il ne reste plus qu'à ouvrir le fichier afin de procéder à l'opération souhaitée (voir les exemples du FMS pour l'ouverture d'un fichier).

2 - Les 2 exemples suivants illustrent la simplicité de l'utilisation des sous-programmes d'E/S fournis par le DØS.

21)	LDAA	# 'A	caractère ASCII A
	JSR	PUTCHR	sortie du caractère
22)	LDX	# STRING	(x) pointe sur une chaîne de caractères
	JSR	PSTRNG	affiche Retour-chariot + Saut ligne + Chaîne

4. LE SYSTEME DE GESTION DES FICHIERS (FMS)

Le système de gestion de fichier (FMS) assure l'interface entre le DØS (système d'exploitation disque) et la mémoire de masse que constituent les disques.

Le FMS gère la création et la destruction de tous les fichiers sur le disque. Tout l'espace est alloué dynamiquement. La zone du disque utilisée par un fichier peut-être réutilisée aussitôt après la destruction de ce fichier.

La communication avec le FMS se fait par l'intermédiaire des FCB (Blocs de contrôle du fichier). Ces blocs contiennent les caractéristiques du fichier comme son nom ou le numéro du disque sur lequel il existe.

La lecture et l'écriture des données d'un fichier (E/S disque) seront faites caractère par caractère, le disque n'étant considéré que comme un terminal du micro-ordinateur.

Les fichiers peuvent être ouverts en lecture ou en écriture. Plusieurs fichiers peuvent être ouverts en même temps du moment qu'ils ont chacun leur FCB.

Le FMS est un langage de commande dans lequel chaque commande est représentée par un nombre appelé : Code Fonction. A chaque Code Fonction correspond une tâche bien déterminée, comme l'ouverture en lecture ou la destruction d'un fichier.

En général, l'utilisation du FMS est relativement simple. Il suffit de pointer le registre d'index sur l'adresse du FCB, de ranger le Code Fonction choisi dans le 1er octet du FCB puis d'appeler le FMS par un JSR ("jump to subroutine").

A aucun moment l'utilisateur n'a à se préoccuper de la place du fichier sur le disque, de sa longueur ou de sa référence dans le catalogue. Le FMS traite ces problèmes automatiquement.

Du fait de la gestion dynamique de l'espace disponible sur le disque par une structure de liste linéaire chaînée, il est possible que des fichiers occupent sur le disque d'une suite des secteurs non contigus.

Normalement, quand une disquette vient d'être formattée, les fichiers qui y seront rangés occuperont des secteurs contigus. Au fur et à mesure des créations et destructions de fichiers, toutefois, le disque peut se "fragmenter" ; c'est-à-dire que les secteurs occupés par un fichier ne se suivent plus physiquement sur le disque : pour le logiciel, ils restent évidemment contigus.

Ceci est une caractéristique des structures de listes linéaires chaînées et des méthodes d'allocation dynamique de l'espace disponible.

L'utilisateur du FLEX n'a pas à se préoccuper de cette fragmentation possible des fichiers ; il doit cependant garder à l'esprit que les fichiers créés peuvent être constitués de secteurs disséminés sur le disque.

La seule conséquence de la "fragmentation" est l'augmentation de la durée de lecture d'un fichier, due à un nombre de déplacements de la tête de lecture plus important.

4.1 - Le bloc de contrôle du fichier

Le bloc de contrôle de fichier (FCB) est le coeur du FMS. Un FCB est un bloc de 320 octets de RAM, dans l'espace mémoire utilisateur, qui est utilisé par les programmes pour communiquer avec le FMS.

A chaque fichier ouvert correspond un FCB particulier. Après la fermeture du fichier, le FCB peut être réutilisé pour ouvrir un autre fichier ou pour exécuter d'autres fonctions comme la destruction (DELETE) ou le changement de nom (RENAME) d'un fichier.

Un FCB peut être localisé n'importe où dans la mémoire utilisateur sauf dans la page 0.

La zone mémoire réservée au FCB n'a pas besoin d'être initialisée de quelque manière que ce soit. Seuls les paramètres utilisés pour l'exécution de la fonction demandée devront être chargés dans le FCB.

Le FMS initialisera lui-même les zones du FCB dont il pourrait avoir besoin.

Dans la description qui suit, le numéro des octets sont désignés par rapport au début du FCB i.e : Octet 0 est le premier octet du FCB.

DESCRIPTION D'UN BLOC DE CONTROLE

- Octet 0 Code fonction

Le code de la fonction choisie doit être stocké dans cet octet, par l'utilisateur, avant d'appeler le FMS. Voir le chapitre de descriptions des codes-fonction.

- Octet 1 Code Erreur

Si une erreur est détectée pendant l'exécution de la fonction, le FMS charge un code erreur dans cet octet et rend la main à l'utilisateur après avoir mis à zéro le bit z (condition zéro) du registre condition du CPU i.e. une condition non-nul existe.

Ceci peut être testé par une instruction BEQ ou BNE.

- Octet 2 Code d'activité

Cet octet est mis à "1" par le FMS si le fichier est ouvert pour lecture et mis à "2" s'il est ouvert pour écriture. Cet octet est testé par plusieurs modules du FMS pour déterminer la légalité de l'opération demandée. Un code erreur est donné en cas d'opération illégale.

Les douze octets suivants (3-14), renferment la "spécification de fichier" qui consiste en un numéro de lecteur, nom de fichier et extension de fichier. Certaines des fonctions du FMS ne requièrent pas le nom du fichier ni son extension.

- Octet 3 Numéro de lecteur

C'est le numéro du lecteur contenant la disquette sur laquelle le fichier se trouve. Ce numéro peut être 0, 1, 2 ou 3.

Manuel de Programmation Avancée

Les 24 octets suivants (4-27) contiennent les informations nécessaires à la dénomination du fichier. Ce sont les mêmes informations que celles contenues dans l'inventaire du disque.

- Octets 4 à 11 Nom du fichier

C'est le nom du fichier choisi. Ce nom doit obligatoirement commencer par une lettre et doit contenir uniquement des lettres, des chiffres, des traits d'union (-) et/ou des soulignés (---).

Si le nom du fichier contient moins de 8 caractères, les octets restants seront à zéro. Le nom du fichier doit être calé à gauche dans le champ qui lui est assigné.

- Octets 12-14 Extension du fichier

C'est l'extension du fichier choisi. Elle doit commencer par une lettre et ne doit contenir que des lettres, des chiffres, des traits d'union (-) et/ou des soulignés (---).

Si la longueur de l'extension est inférieure à 3 caractères, les octets restants seront mis à zéro.

L'extension doit être calée à gauche dans le champ qui lui est assigné. Les fichiers sans extension ne pourront être créés.

- Octet 15 Attributs du fichier

A présent, seuls les 4 bits les plus significatifs sont définis. Ces bits sont utilisés pour définir les types de protection nécessaires. Ils sont définis comme :

bit 7 : protection en écriture
bit 6 : protection contre la destruction
bit 5 : protection en lecture
bit 4 : protection de catalogue

Mettre à "1" l'un de ces bits, met en place la protection qui lui est associée.

Tous les bits sans fonction doivent être à zéro !

- Octet 16 Réservé pour une utilisation future

- Octet 17-18 Adresse de début du fichier sur le disque

Ces deux octets contiennent respectivement le numéro de la piste et du secteur, du premier secteur du fichier.

- Octet 19-20 Adresse de fin du fichier sur le disque

Ces deux octets contiennent respectivement le numéro de la piste et du secteur, du dernier secteur du fichier.

- Octets 21-22 Taille du fichier

C'est un nombre sur 16 bits qui indique le nombre de secteurs occupés par le fichier.

- Octet 23 Indicateur d'une table des secteurs du fichier

Si cet octet est non nul (habituellement 02), le fichier est un fichier à accès direct et contient une table des secteurs du fichier. Voir la description des fichiers à accès direct pour plus de détails.

- Octet 24 Réservé pour une utilisation future

- Octet 25-27 Date de création du fichier

Ces 3 octets contiennent la date, en binaire, de la création du fichier. Le premier octet est le mois, le second le jour, et le troisième l'année (uniquement les dizaines et les unités).

- Octet 28-29 Pointeur de liste des FCB

Tous les FCB qui sont ouverts pour une lecture ou une écriture sont chaînés ensemble. Ces deux octets contiennent l'adresse mémoire du pointeur de liste des FCB du bloc suivant dans la chaîne. Le 1er FCB dans la chaîne est adressé par le pointeur de base des FCB (cf. les variables globales). Ces deux octets sont à zéro si ce FCB est le dernier de la chaîne.

- Octets 30-31 Position courante

Ces octets contiennent respectivement les numéros de la piste et du secteur, du secteur alors contenu dans la zone tampon du FCB. Si le fichier est à écrire, le secteur adressé par ces octets n'a pas encore été écrit sur la disquette ; il est encore dans le buffer.

- Octets 32-33 Numéro d'enregistrement courant

Ces octets contiennent le numéro logique de l'enregistrement du secteur contenu dans le FCB.

- Octet 34 Indice des données

Cet octet contient l'adresse du prochain octet de données à aller chercher (pour lecture) ou à ranger (pour écriture) dans le tampon. Cette adresse est relative au début du tampon; elle est automatiquement avancée par la fonction "Lecture/Ecriture du Prochain Octet". Le programme utilisateur n'a pas besoin d'utiliser cet octet.

- Octet 35 Indice d'accès direct

Cet octet est utilisé conjointement avec la fonction d'accès direct à un octet à l'intérieur d'un secteur, pour lire un octet dans le tampon sans avoir à lire tous les autres octets le précédant.

Manuel de Programmation Avancée

L'adresse de l'octet choisi, calculée relativement au début du secteur, est rangée par l'utilisateur dans l'octet d'indice et la fonction d'Accès Direct à un octet est lancée par le FMS. L'octet de données désigné est retourné dans l'accumulateur A.

Si la valeur rangée dans l'octet d'indice d'accès direct est inférieure à 4, l'utilisateur a accès à l'un des octets de chainage du secteur. Les données proprement dites commencent à une adresse relative supérieure à 4.

- Octets 36-46 Tampon

Ces octets sont utilisés par le FMS pour ranger temporairement le nom d'un fichier. Ces positions ne sont pas utilisées par un programme utilisateur.

- Octets 47-49 Adresse courante du catalogue

Si le FCB est utilisé pour traiter des informations du catalogue. à l'aide de la fonction "Acquisition/Sauvegarde des informations d'un enregistrement", ces trois octets contiennent les numéros de la piste, du secteur et l'indice des données du catalogue contenues dans la zone dénomminative du fichier (octets 4 à 27). Les valeurs de ces trois octets sont remis à jour par la fonction "Acquisition des informations d'un enregistrement".

- Octets 50-52 Pointeur de recherche du 1er fichier effacé dans le catalogue

Ces octets sont utilisés par le FMS lorsqu'il cherche une entrée libre dans le catalogue pour y ranger le nom d'un nouveau fichier.

- Octets 53-63 Zone de travail

Ces octets sont utilisés pour ranger le nouveau nom et l'extension d'un fichier dont on change le nom. Ce nouveau nom répond aux mêmes caractéristiques que celles décrites auparavant (octets 4 à 14).

- Octet 59 Indicateur de compression de l'espace

Si un fichier est ouvert pour lecture ou pour écriture, cet octet indique si une compression d'espace a été faite.

S'il est à zéro, la compression sera faite lors de la lecture ou de l'écriture des données. Cette valeur y est rangée par les opérations "Ouverture pour Lecture" et "Ouverture pour Ecriture" d'un fichier.

S'il est à \$FF, il n'y a pas de compression à faire. Cette valeur doit être rangée par l'utilisateur, après l'ouverture d'un fichier, s'il ne désire pas une compression d'espace (par exemple pour un fichier binaire).

Une valeur positive non nulle indique que la compression est en cours ; cette valeur représente le nombre d'espaces traités jusque là. (Notons que bien que cet octet soit inclus dans la zone de travail décrite ci-dessus, cela n'entraîne aucun conflit dans la mesure où la compression ne peut se faire que si le fichier est ouvert et que la zone de travail n'est utilisée que pour un changement de nom (RENAME), qui ne se fait que fichier fermé).

En règle générale cet octet sera à 0 pour tout travail sur des fichiers TXT et à \$FF pour tout travail sur des fichiers binaires.

- Octets 63-319 Tampon secteur

Ces octets contiennent les données du secteur lu ou écrit. Les 4 premiers octets du secteur sont utilisés par le système, les 252 autres sont utilisés pour y ranger les données.

4.1 bis - Points d'entrée du système de gestion de fichier

1 - Initialisation du système : \$D400

Ce point d'entrée est utilisé par le système d'exploitation disque pour initialiser le FMS après un départ à froid. L'utilisateur ne devrait pas avoir à utiliser ce point d'entrée. Une initialisation au mauvais moment peut détruire les fichiers de données, nécessitant alors une ré-initialisation des disquettes.

2 - Fermeture du système : \$D403

Ce point d'entrée est utilisé par le système d'exploitation disque à la fin de chaque ligne de commande pour fermer tout fichier laissé ouvert par le processeur. Les programmes-utilisateur peuvent aussi utiliser ce point d'entrée pour fermer tous les fichiers ouverts ; toutefois, si une erreur est détectée en essayant de fermer un fichier, tous les fichiers restants resteront dans leur état. Le programmeur est ainsi prévenu contre l'usage de cette routine au lieu de la bonne méthode qui consiste à fermer les fichiers individuellement. Il n'y a pas d'arguments dans cette routine. Elle est entrée par une instruction JSR comme une sous-routine. En sortie, le bit Z du registre de conditions du CPU est à 0 si une erreur est détectée, à 1 sinon ; dans le cas d'une erreur, le registre d'index contient l'adresse du FCB qui cause l'erreur.

3 - Appel du système : \$D406

Ce point d'entrée est utilisé pour tous les autres appels au système de gestion des fichiers.

Un code fonction est rangé dans l'octet n° 0 du FCB, l'adresse du FCB est mis dans le registre d'index du CPU, et ce point d'entrée est appelé par un JSR.

Les différentes fonctions sont décrites ci-dessous.

En sortie, le bit Z du registre de conditions du CPU est mis à "1" si aucune erreur n'a été détectée au cours de l'exécution de la fonction choisie. Ce bit pourra être testé par une instruction BEQ ou BNE. Si une erreur est détectée, le bit Z est mis à zéro et l'octet "code erreur" du FCB est chargé avec le numéro de l'erreur détectée.

Dans tous les cas, le registre d'index du CPU contient l'adresse du FCB.

Quelques fonctions ont besoin de certains paramètres, chargés dans les accumulateurs A et/ou B du CPU.

Pour plus de détails, se référer à la description des différentes fonctions possibles.

Dans tous les cas, le registre d'index X et le registre B ne sont pas modifiés par un appel au FMS.

4.2 - Variables globales

Ce paragraphe décrit les différentes variables, internes au FMS, susceptibles d'intéresser l'utilisateur. Toutes les autres positions mémoire dans la zone du FMS ne doivent pas être utilisées par le programme utilisateur pour y ranger des données.

- \$D409 \$D40A Pointeur de base du FCB

Ces deux octets contiennent l'adresse du pointeur de la liste des FCB dans le premier bloc de la chaîne des fichiers ouverts.

Le contenu de ces octets est géré par le FMS, et le programmeur ne doit y ranger aucune valeur.

Si pour une raison ou une autre, l'utilisateur désire chaîner les blocs de contrôle des fichiers à ouvrir, il doit se souvenir qu'à cette adresse, se trouve le point de départ de la chaîne.

Attention ! il ne faut pas oublier que cette adresse désigne les emplacements dans le FCB du pointeur de liste FCB et non pas le premier mot du FCB.

Si ces deux octets sont nuls, cela signifie qu'il n'y a pas de fichiers à ouvrir.

- \$D40B \$D40C Adresse du bloc de contrôle courant

Ces deux octets contiennent l'adresse du dernier bloc de contrôle traité par le FMS. Cette adresse est celle du premier mot du FCB.

- \$D435 Indicateur de vérification

Si cet octet est non nul, le FMS testera chaque secteur juste après l'écriture afin de détecter d'éventuelles erreurs. Si cet octet est nul, aucun test de vérification à l'écriture ne sera exécuté.

La valeur donnée par défaut à cet octet est "nul".

4.3 - Codes fonction du FMS

L'utilisation du système de gestion des fichiers du FLEX passe par l'utilisation d'un certain nombre de codes fonction.

Le code fonction choisi, est placé par l'utilisateur dans l'octet de code-fonction (octet 0) du bloc de contrôle du fichier (FCB) avant l'appel du FMS.

Le FMS sera appelé par un JSR au point d'appel du FMS (\$D406). Le registre d'index du CPU devra contenir l'adresse du bloc de contrôle.

En sortie du FMS, le bit Z du registre de conditions du CPU sera mis à zéro si une erreur a été détectée en cours d'exécution de la fonction choisie. Ce bit pourra être testé par les instructions BNE et BEQ.

N.B : Dans tous les exemples qui suivent, la ligne "JSR FMS" fait référence au point d'appel du FMS : \$D406.

- Fonction 0 : Lecture/écriture du prochain octet/caractère

Si le fichier est ouvert pour lecture, le prochain octet est lu dans le fichier et transmis au programme appelant dans l'accumulateur A.

Si le fichier est ouvert pour écriture, le contenu de l'accumulateur A est placé dans le buffer afin d'être écrit dans le prochain octet du fichier.

L'indicateur de compression d'espace doit contenir les valeurs convenables pour réaliser ladite compression, si nécessaire (voir dans la description d'un FCB, le paragraphe traitant de l'indicateur de compression d'espace).

En sortie, l'octet de code fonction demeurera inchangé.

Ainsi des lectures/écritures consécutives pourront être exécutées sans avoir à recharger le code fonction.

En lecture, une erreur "Fin de Fichier" est envoyée quand toutes les données du fichier ont été lues.

Quand le secteur courant à lire est vide, le prochain dans le fichier est automatiquement préparé pour l'exécution de la fonction, sans aucune intervention de l'utilisateur.

De même, en écriture, des secteurs pleins sont automatiquement écrits sur le disque sans intervention de la part de l'utilisateur.

Exemple :

En lecture :

```
LDX  ## FCB   Pointe sur l'adresse du FCB
JSR   FMS       Appel du FMS
BNE   ERRR     Test d'erreur
Le caractère lu est maintenant dans l'accu. A.
```

En écriture :

```
LDAA  CHAR      Chargement du caractère
LDX   ## FCB   Pointe sur l'adresse du FCB
JSR   FMS       Appel du FMS
BNE   ERRR     Test d'erreur
Le caractère dans A a été écrit.
```

Fonction 1 Ouverture pour lecture

Le fichier désigné dans le FCB est ouvert pour lecture seulement. Si le fichier ne peut être trouvé, un code erreur est renvoyé. Les zones du FCB qui doivent être initialisées avant l'exécution de la fonction sont le numéro du lecteur, le nom du fichier, l'extension du fichier et le code fonction. Les autres zones du FCB seront initialisées par l'exécution de l'ouverture du fichier.

L'ouverture du fichier met l'indicateur de compression d'espace à zéro, désignant par là un fichier source (TXT). Si le fichier est un fichier binaire, le programmeur devra positionner l'indicateur de compression à \$FF après avoir ouvert le fichier, pour ne pas valider le mode de compression de place.

En sortie du FMS, après l'ouverture d'un fichier, le code fonction dans le FCB du fichier est mis automatiquement à zéro (Lecture/Ecriture du prochain caractère) en anticipation d'une E/S sur le fichier.

Exemple :

```
LDX  ## FCB   Pointe sur l'adresse du FCB
(initialisation des attributs du fichier dans le FCB)
LDAA  ## 1     Code Fonction Ouverture en Lecture
STAA  O,X      Sauvegarde dans le FCB
JSR   FMS      Appel du FMS
BNE   ERRR     Test d'erreur
Le fichier est maintenant ouvert pour lecture.
Dans le cas du fichier binaire, il faut continuer avec :
LDAA  $FF      Suppression compression de place
STAA  59,X     sauvegarde dans le FCB
```

Fonction 2 Ouverture pour écriture

C'est le même type de fonction que la fonction 1 sauf que le fichier ne doit pas déjà exister sur la disquette, et qu'il est ouvert en écriture seulement.

Un fichier ouvert en écriture ne peut être lu tant qu'il n'a pas été fermé puis rouvert en lecture.

L'indicateur de compression de place pourra être manipulé de la même manière qu'il est décrit plus haut dans l'ouverture d'un fichier pour lecture.

Un fichier est normalement ouvert comme un fichier à accès séquentiel mais peut l'être comme un fichier à accès direct en chargeant une valeur non nulle dans l'octet du FCB indicateur d'une table des secteurs, juste après l'ouverture du fichier pour écriture.

Pour plus de détail, se référer aux chapitre traitant des fichiers à accès direct.

Le fichier sera créé sur le disque désigné à moins que le numéro de lecteur soit \$FF. Dans ce cas, le fichier sera créé sur le premier disque prêt (mode recherche automatique).

Exemple :

```
LDX . # FCB    pointe sur l'adresse du FCB
                (initialisation des attributs du fichier dans le FCB)
LDAA # 2        Code fonction Ouverture en Ecriture
STAA 0,X        Sauvegarde dans le FCB
JSR  FMS        Appel du FMS
BNE  ERRØR      Test d'erreur
Le fichier est maintenant ouvert pour écriture.
```

Dans le cas d'un fichier binaire, il faut continuer avec :

```
LDAA $FF        Suppression compression de place
STAA 59,X
```

- Fonction 3 : Ouverture pour mise à jour

Cette fonction permet d'ouvrir un fichier aussi bien pour lecture qu'écriture. Le fichier ne doit pas être déjà ouvert et doit exister sur le disque spécifié. Si la spécification de lecteur est \$FF, tous les lecteurs seront recherchés.

Une fois que le fichier est ouvert pour être remis à jour, quatre opérations peuvent être exécutées :

- 1 - Lecture en accès séquentiel
- 2 - Lecture en accès direct
- 3 - Ecriture en accès direct
- 4 - Fermeture du fichier

Il n'est pas possible d'écrire séquentiellement sur un fichier ouvert pour une remise à jour. Cela implique, qu'il n'est pas possible d'augmenter la taille d'un fichier à l'aide de la fonction 3.

Manuel de Programmation Avancée

Fonction 4 : Fermeture d'un fichier

Si le fichier est ouvert pour lecture, sa fermeture supprime simplement son bloc de contrôle de la chaîne des FCB. Si le fichier est ouvert pour écriture, les données restantes dans le tampon secteur sont écrites sur le disque, en y ajoutant des 0 si besoin est pour remplir le secteur.

Si un fichier ouvert pour écriture n'est jamais écrit, son nom ne sera pas inscrit dans le catalogue du disque puisqu'il ne contient pas de données.

Exemple :

```
LDX  # FCB   Pointe sur l'adresse du FCB
LDAA # 4      Code fermeture
STAA 0,X      Sauvegarde dans le FCB
JSR  FMS      Appel du FMS
BNE  ERROR    Test d'erreur
Le fichier est maintenant fermé.
```

Fonction 5 : Repositionnement au début d'un fichier

Seuls les fichiers ouverts pour lecture peuvent être repositionnés. En sortie du FMS, le code fonction dans le FCB est mis à 0, anticipant un ordre de lecture du fichier. Si le programmeur désire repositionner au début un fichier ouvert pour écriture, de telle sorte qu'on puisse maintenant le lire, il doit d'abord le fermer puis le rouvrir pour lecture.

Exemple :

```
Le fichier est déjà ouvert pour lecture
LDX  # FCB   Pointe sur l'adresse du FCB
LDAA # 5      Code de repositionnement
STAA 0,X      sauvegarde dans le FCB
JSR  FMS      Appel du FMS
BNE  ERROR    Test d'erreur
Le fichier est maintenant repositionné au début et prêt en lecture.
```

Fonction 6 : Ouverture du catalogue

Cette fonction permet l'ouverture du catalogue afin qu'un programme puisse y accéder. Le FCB utilisé pour réaliser cette fonction ne doit pas l'être déjà pour l'utilisation d'un fichier.

En entrée, la seule information à stocker dans le FCB est le numéro du lecteur. Aucun nom de fichier n'est nécessaire. Les informations dans le catalogue sont lues en utilisant la fonction "acquisition des données d'un enregistrement".

La fonction "écriture des données d'un enregistrement" est utilisée pour écrire dans le catalogue.

La fonction "Lecture/Ecriture du prochain caractère" ne fonctionne pas sur un FCB ouvert pour accéder au catalogue.

Il n'est pas nécessaire de fermer un FCB ouvert par la fonction 6 après avoir terminé la manipulation sur le catalogue. Normalement, l'utilisateur ne doit pas avoir besoin d'utiliser cette fonction.

Fonction 7 : Acquisition des données d'un enregistrement

Cette fonction doit être utilisée, uniquement, avec un FCB ouvert pour "accès catalogue" (fonction 6). Chaque fois qu'elle est terminée, le prochain enregistrement dans le catalogue est chargé dans la zone des "informations du catalogue" du FCB (octets 4 à 27). Tous les enregistrements du catalogue, y compris ceux qui ne sont pas utilisés, ou ceux qui sont effacés, sont lus lorsque cette fonction est utilisée.

Après la lecture d'un enregistrement, le FCB "pointe" sur l'enregistrement qui vient d'être lu ;

L'adresse courante du catalogue dans le FCB (octets 47-49) est celle de l'enregistrement qui vient d'être lu.

Une erreur "Fin de Fichier" est retournée lorsque la fin du catalogue est atteinte.

Exemple :

Acquisition du 3ème enregistrement dans le catalogue.

```

LDX  ## FCB      pointe sur l'adresse du FCB
LDAA  DISQUE      (DISQUE) = n° du lecteur choisi
STAA  3,X         Sauvegarde dans le FCB
LDAA  ## 6        Code ouverture du catalogue
STAA  0,X         Sauvegarde dans le FCB
JSR   FMS         Appel du FMS
BNE   ERROR       Test d'erreur
LDAB  ## 3        Init. compteur à 3
LØØP LDAA ## 7      Code acquisition d'un enregistrement
STAA  0,X         Sauvegarde dans le FCB
JSR   FMS         Appel du FMS
BNE   ERROR       Test d'erreur
DECB                      Décrémente compteur
BNE   LØØP        répété jusqu'à la fin
Le 3ème enregistrement du catalogue est maintenant dans le FCB.
```

Fonction 8 : Ecriture des données d'un enregistrement

Cette fonction doit être utilisée uniquement avec un FCB ouvert pour "accès catalogue". L'enregistrement à écrire dans le catalogue est recopié de la "zone des informations du catalogue" du FCB dans le catalogue, à l'endroit désigné par le FCB. Le secteur du catalogue venant d'être mis à jour est alors automatiquement récrit sur la disquette, pour être certain que le catalogue a bien été remis à jour.

Un programme utilisateur, ne doit, normalement, pas avoir à écrire dans le catalogue.

Il faut utiliser, cette fonction avec précaution. Une erreur de manipulation peut détruire les fichiers sur la disquette.

Fonction 9 : Lecture d'un secteur

Cette fonction permet la lecture d'un secteur sur le disque. Ce secteur est adressé par les octets 30-31 du FCB (position courante). Les données lues sont chargées dans le tampon secteur du FCB.

Cette fonction est normalement exécutée par le FLEX et l'utilisateur ne devrait pas avoir besoin de la programmer. Il est préférable d'utiliser la fonction "Lecture/Ecriture du prochain caractère" chaque fois que possible.

Cette fonction doit être manipulée avec prudence, car sa procédure ne ressemble pas à celle des autres fonctions du FLEX.

Exemple :

```
LDX  #FCB    pointe sur l'adresse du FCB
LDAA  PISTE   (PISTE) = n° de la piste choisie
STAA  30,X    init. FCB position courante
LDAA  SECT    (SECT) = n° du secteur choisi
STAA  31,X    init. FCB position courante
LDAA  #9      Code Lecture d'un secteur
STAA  0,X     Sauvegarde dans le FCB
JSR   FMS     Appel du FMS
BNE   ERRØR   Test d'erreur
```

Le contenu du secteur est maintenant dans le FCB.

Fonction 10 : Ecriture d'un secteur (\$0A hex.)

Cette fonction permet d'écrire directement un secteur sur le disque. Il faut faire très attention en l'utilisant. Cette fonction est normalement exécutée par FLEX et l'utilisateur ne devrait pas avoir à la programmer. Il est préférable d'utiliser la fonction "Lecture/Ecriture du prochain caractère" chaque fois que possible.

Une erreur dans l'utilisation de cette fonction peut provoquer la destruction de la disquette nécessitant sa ré-initialisation.

L'adresse du secteur à écrire doit être contenue dans les octets 30-31 du FCB (position courante) et les données à écrire doivent être présentes au tampon secteur du FCB. Cette fonction teste l'indicateur de vérification (voir le chapitre sur les variables globales) et vérifiera le contenu du secteur sur le disque après l'écriture si le mode de vérification est programmé.

Fonction 11 : Réservé pour une utilisation future (\$0B hex.)

Fonction 12 : Destruction d'un fichier (\$0C hex.)

Cette fonction permet d'effacer le fichier dont le nom est contenu dans le FCB (n° du lecteur, nom du fichier, extension). Les secteurs utilisés par ce fichier peuvent être immédiatement réutilisés.

Le fichier ne doit pas être ouvert lorsque cette fonction est demandée.

Le nom du fichier dans le FCB est altéré pendant l'exécution de cette fonction.

Exemple :

```
LDX  # FCB   pointe sur l'adresse du FCB
      (mise en place du nom du fichier dans le FCB)
LDAA # 12     code destruction
STAA 0,X      Sauvegarde dans le FCB
JSR  FMS      Appel du FMS
BNE  ERROR    Test d'erreur
Le fichier est maintenant effacé.
```

Fonction 13 : Changement du nom d'un fichier (\$0D hex.)

En entrée, le fichier ne doit pas être ouvert. L'ancien nom doit être dans la zone d'identification du fichier du FCB (octets 4 à 27) et le nouveau nom dans la zone de travail du FCB (octets 53 à 63).

Le fichier dont le nom est contenu dans la zone d'identification du FCB porte maintenant le nom contenu dans la zone de travail.

Le nouveau nom et la nouvelle extension du fichier doivent être obligatoirement spécifiés.

Exemple :

```
LDX  # FCB   pointe sur l'adresse du FCB
      (mise en place du nom du fichier dans la zone d'identification)

LDAA # 13     code ré-appellation
STAA 0,X      sauvegarde dans le FCB
JSR  FMS      appel du FMS
BNE  ERROR    Test d'erreur
```

Le fichier est maintenant renommé.

Manuel de Programmation Avancée

Fonction 14 : Réservé pour une utilisation future (\$0E hex.)

Fonction 15 : Accès séquentiel au prochain secteur (\$0F hex.)

En entrée, le fichier peut être ouvert pour lecture ou pour écriture (pas pour une mise à jour).

Si le fichier est ouvert pour lecture, lors de l'exécution de cette fonction, tous les octets de données restants (i.e. pas encore lus) sont ignorés et le pointeur de données sera positionné sur le premier octet de données du secteur suivant dans le fichier. Si le fichier est ouvert pour écriture, au cours de l'exécution de cette fonction la partie non écrite du secteur courant sera initialisée à ZERO, puis le secteur entier sera écrit sur le disque. Le caractère suivant, à écrire dans ce fichier, sera placé dans la première position de la zone des données du secteur suivant. Il est à noter que tous les appels à cette fonction seront ignorés tant qu'il n'y aura pas un octet d'écrit ou de lu dans le secteur courant.

Fonction 16 : Ouverture de l'enregistrement des informations système (\$10 hex.)

En entrée, seul le n° du lecteur doit être spécifié dans le FCB ; il n'y a pas de nom de fichier associé.

Le FCB choisi ne doit pas être déjà ouvert pour l'utilisation d'un fichier.

Cette fonction permet l'accès à l'enregistrement des informations système de la disquette désignée dans le FCB.

Il n'y a pas de fonctions différentes pour la lecture ou la modification de ce secteur. Toutes les manipulations sur les données contenues dans l'enregistrement des informations système, sont faites directement dans le tampon secteur du FCB. Il n'est pas nécessaire de fermer le FCB après exécution de cette fonction.

Cette fonction est utilisée de manière interne par le FLEX ; elle ne devrait normalement pas être appelée par un programme-utilisateur.

Une mauvaise utilisation peut avoir pour résultat, la destruction du contenu de la disquette.

Fonction 17 : Acquisition d'un octet du secteur par accès direct (§11 hex.)

En entrée, le fichier doit être ouvert pour lecture ou pour mise à jour. Le numéro de l'octet désiré doit être rangé dans l'octet "indice d'accès direct" du FCB. (octet 35) Ce numéro est relatif au début du tampon secteur.

En sortie, l'octet choisi est retourné au programme appelant dans l'accumulateur A.

L'"indice d'accès direct" ne doit pas être inférieur à 4 car il n'y a pas de données utilisateur dans les 4 premiers octets du secteur.

exemple :

Lecture du 54ème octet de donnée

LDX	~~#~~ FCB	pointe sur l'adresse du FCB
LDAA	~~#~~ 54 + 4	init. numéro de l'octet choisi
STAA	35,X	sauvegarde dans le FCB
LDAA	~~#~~ 17	code fonction
STAA	0,X	sauvegarde dans le FCB
JSR	FMS	appel du FMS
BNE	ERRØR	test d'erreur

Maintenant, le caractère choisi est dans l'accu. A

Fonction 18 : Ecriture d'un octet dans le secteur par accès direct (§12 hex.)

Le fichier doit être ouvert pour mise à jour. Cette fonction est semblable, à la précédente, mis à part que le caractère chargé dans l'accumulateur A est écrit dans le secteur à la position indiquée par l'octet d'"indice d'accès direct" du FCB.

L'"indice d'accès direct" ne doit pas être plus petit que 4 car les 4 premiers octets de chaque secteur sont réservés à des données système.

Exemple :

Ecriture d'une donnée dans le 54ème octet du secteur courant

LDX	~~#~~ FCB	pointe sur l'adresse du FCB
LDAA	~~#~~ 54 + 5	init. numéro de l'octet choisi
STAA	35,X	sauvegarde dans le FCB
LDAA	~~#~~ 18	code fonction
STAA	0,X	sauvegarde dans le FCB
LDAA	CARACT.	acquisition du caractère à écrire
JSR	FMS	appel du FMS
BNE	ERRØR	test d'erreur

Le caractère a été écrit dans le secteur.

Fonction 19 : Réservée pour une utilisation future (\$13 hex.)

Fonction 20 : Recherche du lecteur suivant (\$14 hex.)

Cette fonction est utilisée pour trouver le prochain lecteur prêt. FLEX exécute cette fonction de manière différente suivant les versions disques 5" ou 8". Cela est dû aux limites du matériel.

Le déroulement de cette fonction est le suivant pour la version 8" :

Si le numéro de lecteur dans le FCB est \$FF (hex), la recherche des lecteurs prêts démarre sur le lecteur 0. Si le numéro du lecteur dans le FCB est 0, 1, ou 2, la recherche démarre par le lecteur 1, 2 ou 3 respectivement. Si un lecteur est prêt, son numéro est chargé dans l'octet "numéro de lecteur" du FCB et le bit de condition 0 (zéro) dans le registre de condition du CPU est mis à "1". Si aucun des lecteurs n'est prêt, le bit Z du registre de conditions du CPU est mis à "0" et l'erreur # 16 (lecteur pas prêt) est positionné.

Dans la version 5", le déroulement de la fonction est le suivant :

Si le numéro de lecteur dans le FCB est \$FF (hex), au retour le numéro de lecteur dans le FCB sera 0. Si cette fonction est appelée alors que le numéro de lecteur dans le FCB est 0, au retour celui-ci sera 1. Dans les deux cas, le bit de condition 0 dans le registre de condition du CPU sera mis à "1".

Si elle est appelée alors que le numéro de lecteur dans le FCB est 1 ou plus grand que 1, le numéro de lecteur reste inchangé, le bit Z du registre de conditions du CPU est mis à "0" et l'erreur # 16 (lecteur non prêt) est positionnée.

Fonction 21 : Accès direct à un enregistrement (\$15 hex.)

C'est l'une des fonctions permettant l'accès direct à un secteur d'un fichier. Le numéro de l'enregistrement désiré est chargé dans le FCB, dans le "numéro d'enregistrement courant" (octets 32-33).

Avant d'utiliser cette fonction, le fichier doit être ouvert pour lecture ou pour mise à jour.

Le premier enregistrement des données d'un fichier est l'enregistrement n°1. Se positionner sur l'enregistrement 0, déclenchera la lecture du premier secteur de la Table des secteurs du fichier. Après s'être correctement positionné sur un enregistrement, le premier caractère lu (accès séquentiel) est le premier octet de données de l'enregistrement spécifié. Essayer d'accéder à un enregistrement qui n'existe pas, déclenche une erreur.

Pour plus d'informations sur les fichiers à accès direct, lire le chapitre intitulé "Fichiers à accès direct".

Exemple :

Positionnement sur l'enregistrement n°6

LDX	# FCB	pointe sur l'adresse du FCB
LDAA	# 6	init. position
STAA	93,X	sauvegarde dans le FCB
CLR	32,X	init M.S.B. position à 0
LDAA	# 21	code fonction
STAA	0,X	sauvegarde dans le FCB
JSR	FMS	appel du FMS
BNE	ERRØR	test d'erreur

L'enregistrement est prêt à être lu.

Fonction 22 : Retour arrière d'un enregistrement (\$16 hex.)

Cette fonction est aussi utilisée pour les fichiers à accès direct. Elle prend le numéro d'enregistrement courant dans le FCB et le décrémente de 1. Un re-positionnement sur le nouvel enregistrement est exécuté. Cela a pour effet de se décaler en arrière d'un enregistrement en entier.

Par exemple, si le numéro d'enregistrement courant est #16 et que la fonction "retour arrière d'un enregistremeent" est exécutée, le fichier sera alors positionné pour permettre la lecture du premier octet de l'enregistrement # 15.

Le fichier doit être ouvert pour lecture ou pour mise à jour avant l'utilisation de cette fonction.

Voir le chapitre "fichiers à accès direct" pour plus de détails.

5 - FICHIERS A ACCES DIRECT

Le Système d'exploitation FLEX permet la gestion des fichiers en accès direct.

La technique de l'accès direct permet d'atteindre n'importe quel secteur d'un fichier en précisant simplement la position correspondant à cet enregistrement. La longueur du fichier n'a pas d'importance, et le temps d'accès au secteur choisi est celui de deux lectures, au maximum.

En utilisant le même mécanisme, l'utilisateur peut aussi aisément se positionner sur le nième caractère d'un secteur. Seul un petit calcul utilisant le nombre d'octets de données dans un secteur (252) est nécessaire.

Tous les fichiers ne peuvent être manipulés en accès direct. Il est nécessaire qu'ils aient été auparavant créés en tant que fichiers à accès direct.

Le mode de création d'un fichier est par défaut le mode séquentiel. Tous les utilitaires standard du FLEX fonctionnent avec des fichiers à accès séquentiel.

Le seul fichier à accès direct utilisé par FLEX est le fichier ~~ERRORS~~.SYS.

Un fichier créé comme un fichier à accès direct peut être lu de manière directe ou séquentielle.

Un fichier créé comme un fichier à accès séquentiel ne peut être lu que de manière séquentielle.

Pour créer un fichier à accès direct, la procédure normale d'ouverture pour écriture doit être utilisée. Puis, tout de suite après l'ouverture (si elle est réussie...), il faut initialiser l'indicateur de table des secteurs du fichier dans le FCB, à une valeur non nulle, ensuite la création du fichier peut commencer.

Seule la création de fichiers texte en accès direct, présente un intérêt.

Lors de la création du fichier le système crée une table des secteurs du fichier.

Manuel de Programmation Avancée

La table des secteurs du fichier (FSM) est une table ou catalogue qui permet au système de repérer chaque enregistrement (secteur) du fichier sur le disque.

La FSM est constituée de 2 secteurs et le numéro d'enregistrement 0 lui est affecté.

Ceci implique qu'un fichier de données, de 5 secteurs de long aura finalement une longueur de 7 secteurs s'il est créé en accès direct.

L'utilisateur n'a pas besoin de manipuler les secteurs de la FSM, aussi lorsqu'un fichier est ouvert en lecture, ils sont automatiquement sautés. Par contre, le FMS les utilise pour exécuter les fonctions 21 et 22.

La zone d'identification d'un fichier, contient l'information précisant si un fichier a été créé en accès direct ou non.

Si l'octet indicateur de table des secteurs d'un fichier est non nul, le fichier est un fichier à accès direct sinon, il est à accès séquentiel.

Il est à noter qu'un fichier à accès direct peut être recopié sur une autre disquette sans perdre ses propriétés, mais qu'il ne peut pas être fusionné à un autre fichier.

6 - ERREURS FLEX

1 - CODE DE FONCTION FMS ILLEGAL

La valeur du code de fonction dans le FCB est trop grande ou interdite. L'erreur se déclenche à l'appel du FMS.

2 - LE FICHIER SPECIFIE EST DEJA OUVERT

Une fonction d'ouverture pour lecture, mise à jour, ou écriture a été programmée dans un bloc de contrôle pour un fichier déjà ouvert.

3 - LE FICHIER SPECIFIE EXISTE DEJA

a) - Une ouverture pour écriture est exécutée avec un FCB contenant les spécifications d'un fichier déjà existant sur la disquette.

b) - La fonction "changement de nom d'un fichier" est demandée alors que le nom existe déjà sur la disquette.

4 - LE FICHIER SPECIFIE N'A PAS ETE TROUVE

L'ouverture d'un fichier pour lecture ou pour remise à jour ou pour ré-appellation ou pour destruction a été demandée alors que le FCB correspondant contient les spécifications d'un fichier n'existant pas sur la disquette.

5 - ERREUR CATALOGUE SYSTEME - RECHARGEZ LE SYSTEME

Réservé pour une utilisation future.

6 - PLUS D'ESPACE DISQUE POUR LE CATALOGUE

Cela signifie que le disque est plein, l'allocation d'espace pour le catalogue étant dynamique.

7 - TOUT L'ESPACE DISQUE A ETE UTILISE

Tout l'espace utile de la disquette a été utilisé par les fichiers. Si cette erreur est renvoyée par le FMS, le dernier caractère qui devait être écrit dans le fichier ne l'a pas été.

8 - FIN DE FICHIER RENCONTRE EN LECTURE

Une opération de lecture d'un fichier s'est terminée par la rencontre d'une Fin de Fichier (E.Ø.F.). Toutes les données du fichier ont été lues. Cette erreur peut être renvoyée au cours de la lecture du catalogue du disque, à l'aide de la fonction : "Acquisition des données d'un enregistrement" (fonction 7), quand la fin du catalogue est atteinte.

9 - ERREUR DE LECTURE SUR DISQUE

Une erreur de parité (ou autre...) a été détectée par le contrôleur au cours d'un essai de lecture d'un secteur.

Le FLEX aura automatiquement essayé de relire le secteur défectueux plusieurs fois sans succès avant de rapporter l'erreur.

Cette erreur peut aussi être due au chargement de numéros de piste et de secteurs illégaux dans le FCB.

10 - ERREUR D'ECRITURE SUR DISQUE

Une erreur de parité (ou autre...) a été détectée par le contrôleur au cours d'un essai d'écriture d'un secteur. Le FLEX aura automatiquement essayé de récrire plusieurs fois le secteur défectueux, sans succès, avant de rapporter l'erreur.

Cette erreur peut aussi être due au chargement de numéros de piste et de secteur illégaux, dans le FCB. Une erreur d'écriture sera aussi renvoyée si une erreur de lecture est détectée par le FLEX au cours de la mise à jour du catalogue de la disquette.

11 - FICHER OU DISQUE PROTEGE EN ECRITURE

Un essai d'écriture a été tenté sur une disquette protégée en écriture (découpe d'autorisation d'écriture sur l'enveloppe du disque) ou sur un fichier alors que le bit de protection d'écriture est positionné à "1".

12 - FICHER PROTEGE - NON EFFACE

Le bit de protection contre la destruction du fichier spécifié est mis à "1" et celui-ci ne peut être détruit.

13 - BLOC CONTROLE DE FICHER ILLEGAL

Une tentative d'accès à un bloc de contrôle dans la chaîne des FCB a été faite, mais celui-ci n'y existe pas.

14 - ADRESSE DISQUE ILLEGALE

Réservé pour une utilisation future.

15 - NUMERO D'UNITE DISQUE ILLEGAL

Réservé pour une utilisation future.

16 - UNITE DISQUE NON PRETE

Le lecteur n'a pas de disquette, ou sa porte n'est pas fermée. Ce message ne peut être sorti s'il n'y a aucun moyen de détecter cet état sur les unités (certains 5" par exemple).

17 - LE FICHIER EST PROTEGE - ACCES REFUSE

Réservé pour une utilisation future.

18 - VIOLATION DES ATTRIBUTS D'ACCES A UN FICHIER

a) - Une lecture ou un repositionnement ont été tentés sur un fichier fermé, ou ouvert pour écriture.

b) - Une écriture a été tentée sur un fichier fermé, ou ouvert pour lecture.

19 - POINTEUR D'ACCES DIRECT ERRONE

La fonction accès direct d'un octet du secteur est appelé avec un pointeur supérieur à 255.

20 - FMS INACTIF - RECHARGEZ LE SYSTEME

Réservé pour une utilisation future.

21 - SPECIFICATION FICHIER ILLEGALE

Une erreur de format a été détectée dans l'appelation d'un fichier.

Le nom du fichier doit commencer par une lettre, et ne doit contenir que des lettres, des chiffres, des traits d'union et/ou des soulignés.

Idem pour l'extension d'un fichier.

Le nom d'un fichier est limité à 8 caractères, l'extension à 3.

22 - ERREUR SYSTEME EN FERMETURE FICHIER

Réservé pour une utilisation future.

23 - DEBORDEMENT TABLE D'ALLOCATION - DISQUE TROP SEGMENTE

Une tentative de création d'un très grand fichier à accès direct, a été faite sur un disque trop segmenté. Toutes les informations sur l'emplacement des enregistrements n'ont pu être rangées dans les 2 secteurs de la table des secteurs du fichier (FSM).

Manuel de Programmation Avancée

Pour résoudre ce problème, il suffit de recréer ce fichier sur une nouvelle disquette préalablement formatée.

24 - NUMERO D'ENREGISTREMENT INEXISTANT

Au cours de l'exécution de la fonction "Accès direct à un enregistrement" (fonction 21), le numéro de l'enregistrement choisi est plus grand que le numéro du dernier enregistrement du fichier.

25 - ERREUR NUMERO D'ENREGISTREMENT - FICHIER ENDOMMAGE

L'enregistrement localisé par le FMS au cours d'un accès direct au fichier, n'est pas le bon. Le fichier est probablement endommagé.

26 - ERREUR DE SYNTAXE - REDONNEZ LA COMMANDE

Une erreur de syntaxe a été détectée dans la ligne de commande venant d'être tapée.

27 - COMMANDE INTERDITE PENDANT L'IMPRESSION

La commande qui vient d'être entrée ne peut pas être exécutée pendant une impression simultanée ("spooling" actif).

28 - CONFIGURATION HARDWARE INSUFFISANTE

Cette erreur est affichée généralement, si la configuration mémoire installée ne suffit pas à l'exécution d'une fonction donnée.

7. INFORMATIONS PARTICULIERES SUR LES DISQUES ET FICHIERS

7.1 - Modules d'interface du disque

Les informations suivantes sont pour les utilisateurs qui souhaitent écrire leurs propres modules d'interface. Aucune garantie ne peut être réclamée en cas de disfonctionnement du FLEX, si les modules d'interface du disque ont été modifiés.

Les modules d'interface du disque, sont des programmes d'interface entre le FLEX et le contrôleur du système disque. Les modules fournis avec FLEX sur GOUPIL sont prévus pour les contrôleurs formateurs Western Digital (série 1771 ou 179X).

Les modules d'interface du disque sont localisés en RAM entre \$DE80 (hex) et \$DFFF (hex). Toutes les fonctions du disque sont des sauts vectorisés au début de cette zone. En cas d'erreur, les modules d'interface du disque n'ont pas à tenter de reprises ; FLEX les gère automatiquement. Il suffit que le code erreur fourni par le contrôleur soit chargé dans le registre B du CPU et que le bit Z du registre de condition du CPU soit mis à "0". Les codes d'erreurs doivent être conformes aux codes Western Digital.

Ces codes doivent donc être reproduits par logiciel si un autre type de contrôleur est utilisé.

Tous les modules d'interface du disque ne modifie pas le contenu du registre d'index du CPU (X).

Tous les modules d'interface du disque sont appelés par une instruction JSR.

Les modules d'interface décrit ci-après sont communs à toutes les versions du FLEX.

\$DE80 - LECTURE

Entrée :

- (X) = adresse du buffer secteur dans le FCB
- (A) = n° de piste
- (B) = n° de secteur.

Le secteur désigné par le numéro de piste et de secteur sur le disque est lu et chargé dans le tampon secteur du FCB indiqué.

\$DE83 - ECRITURE

Entrée :

- (X) = adresse du buffer secteur dans le FCB
- (A) = n° de piste
- (B) = n° de secteur.

Le contenu du tampon secteur du FCB indiqué est écrit sur le disque, dans le secteur désigné par les numéros de piste et de secteur.

\$DE86 - VERIFICATION

Entrée : pas de paramètres.

Le secteur qui vient d'être écrit est vérifié pour repérer d'éventuelles erreurs CRC.

\$DE89 - REPOSITIONNEMENT SUR LA PISTE 00

Entrée : (X) = adresse du FCB.

Sortie : bit C = 0, bit Z = 1 et (B) = \$0B si le disque est protégé en écriture

bit C = 1, bit Z = 1 et (B) = \$0F s'il n'y a pas de disque.

Une opération de repositionnement sur la piste 00 est exécutée sur le disque dont le numéro est contenu dans le bloc de contrôle du fichier (FCB) désigné.

\$DE8C - SELECTION DU DISQUE

Entrée : (X) = adresse du FCB.

Le disque dont le numéro est contenu dans le bloc de contrôle du fichier sera sélectionné.

\$ E8F - TEST DISQUE PRET

Entrée : (X) = adresse du FCB

Sortie : bit Z = 1 et bit C = 1 si le disque n'est pas prêt

bit Z = 0 et bit C = 0 si le disque est prêt.

Ce module existe s'il est possible de tester l'état du disque dont le numéro est précisé dans le FCB, après avoir sélectionné ce disque et attendu le temps nécessaire à la stabilisation de la vitesse des moteurs (1 seconde environ dans le cas des 5"). Dans les configurations qui ne permettent pas ce test, ce module doit simplement renvoyer l'état prêt si le numéro du disque est 0 ou 1 dans le FCB, et un état pas prêt pour tout autre numéro ;

\$DE92 - TEST RAPIDE DISQUE PRET

7.2 - Initialisation des disquettes

La commande NEWDISK est utilisée pour "initialiser" une disquette suivant un format reconnu par FLEX.

Le processus d'initialisation consiste en l'écriture des numéros de piste et de secteurs sur chaque secteur d'une disquette, suivant le format utilisé par FLEX. En plus, ce processus relie ensemble tous les secteurs de la disquette pour former une chaîne de secteurs disponibles.

La première piste de la disquette, piste 00, est spéciale.

Aucun des secteur de cette piste n'est autorisée à recevoir des fichiers de données. Ils sont réservés à l'usage du système FLEX.

Les deux premiers secteurs contiennent un "chargeur binaire" qui est appelé par la commande  du moniteur GPMON. Ce "chargeur binaire", une fois résident en mémoire centrale charge FLEX de la disquette vers le système.

Le secteur 3 sur cette piste contient l'enregistrement des informations système. Ce secteur contient les adresses physiques sur le disque, de début et de fin des secteurs de la chaîne des secteurs libres, réservés aux fichiers de données.

Tout le reste des secteurs à partir du 5 de la piste 00, est réservé au catalogue des noms de fichiers.

Après l'initialisation, toutes les pistes disponibles ont un format commun.

Les deux premiers octets de chaque secteur contiennent le numéro de la piste et du secteur du prochain secteur de la chaîne. Les deux octets suivants seront utilisés pour y charger le numéro logique du secteur dans le fichier. Les 252 octets restants sont à 0.

A l'origine tous les octets représentent les numéros d'enregistrement sont à 0.

Quand des données sont stockées dans un fichier, les deux octets de lien au début de chaque secteur sont modifiés pour pointer sur le secteur suivant dans le fichier et non plus dans la chaîne des secteurs disponibles.

Les secteurs de l'inventaire du disque sur la piste 00 ont aussi des octets de lien de même type que ceux décrits dans la chaîne des secteurs disponibles et dans les fichiers de données.

Manuel de Programmation Avancée

Une disquette FLEX n'est pas initialisée dans un format IBM standard. En effet, dans un format standard, tous les secteurs se suivent physiquement dans l'ordre défini par la suite logique de leur numéro.

i.e. Le secteur 2 suit le secteur 1, le secteur 3 suit le secteur 2 etc...

Sur une disquette FLEX les secteurs sont entrelacés. Si les secteurs se suivaient physiquement, à la lecture d'un enregistrement, après l'acquisition des données, le secteur suivant directement celui-ci serait déjà passé sous la tête de lecture avant que le deuxième ordre de lecture ne soit donné. Aussi l'entrelaçage des secteurs accorde-t-il plus de temps pour manipuler les données.

Le phénomène de "perte" d'un secteur du aux temps de traitement relativement élevés est appelé "révolutions perdues", et a pour résultat de ralentir considérablement l'exécution des programmes utilisant des fichiers.

Le format du FLEX permet de réduire le nombre de révolutions perdues et donc de diminuer le temps d'exécution d'un programme.

7.3 - Description d'un secteur du catalogue

Chaque secteur dans la partie de la disquette FLEX réservée au catalogue, est composé de 10 "enregistrements catalogue". Chaque enregistrement correspond à un fichier sur la disquette. Dans chaque secteur, les quatre premiers octets contiennent les informations de chainage, et les 12 suivants ne sont pas utilisés.

Si on utilise la fonction "Acquisition des données d'un enregistrement" pour lire les informations contenues dans le catalogue, ces 16 octets sont automatiquement sautés.

Chaque enregistrement du catalogue contient exactement les mêmes informations que celles stockées dans la zone d'identification du fichier dans les FCB (octets 4-27).

Pour plus de détails, se référer à la description du bloc de contrôle du fichier (FCB).

Un enregistrement du catalogue qui n'a jamais été utilisé a un 0 dans le premier octet du nom du fichier.

Un enregistrement du catalogue qui a été effacé a le bit de poids le plus fort du premier octet du nom du fichier mis à "1" (i.e. le premier octet du nom du fichier est négatif).

DESCRIPTION D'UN SECTEUR DE DONNEES

Chaque secteur sur une disquette FLEX (sauf les deux secteurs du "chargeur binaire") a le format suivant :

- Octets 0-1 adresse du prochain secteur
- Octets 2-3 numéro logique du secteur dans le fichier
- Octets 4-255 données

Si un fichier est composé de plus d'un secteur, les octets "d'adresse du prochain secteur", contiennent les numéros de la piste et du secteur, respectivement, du prochain secteur dans le fichier. Ces octets sont à zéro, si c'est le dernier secteur du fichier (Condition E.Ø.F. : fin de fichier).

L'utilisateur ne doit jamais changer les octets de chainage d'un secteur, ceux ci étant automatiquement gérés par le FMS.

7.4 - Description d'un fichier binaire

Sous FLEX, un fichier binaire peut contenir n'importe quel type de données ; tous les caractères ASCII sont autorisés. Chaque fichier binaire est composé d'un ou de plus d'un enregistrement binaire. Il peut y avoir plus d'un enregistrement binaire dans un secteur.

Un fichier binaire a le format suivant :

- Octet 0 Indicateur de début d'enregistrement (\$02, STX)
- Octet 1 MSB de l'adresse de chargement
- Octet 2 LSB de l'adresse de chargement
- Octet 3 Nombre d'octets de données dans l'enregistrement
- Octet 4-n Données binaires de l'enregistrement.

L'adresse de chargement représente l'adresse de la zone mémoire où les données étaient stockées avant d'être écrites sur disque (commande : SAVE).

Quand un fichier binaire est chargé en mémoire à partir du disque, il sera rangé dans la même zone mémoire que celle d'où il est issu.

Un fichier binaire peut aussi contenir une adresse de transfert (adresse d'exécution) facultative dans un enregistrement. Cet enregistrement donne alors l'adresse mémoire du point d'entrée du fichier binaire.

Le format d'un enregistrement contenant une adresse de transfert est le suivant :

- Octet 0 Indicateur d'adresse de transfert (\$16.AC4)
- Octet 1 MSB de l'adresse de transfert
- Octet 2 LSB de l'adresse de transfert.

Si un fichier binaire contient plus d'un enregistrement possédant une adresse de transfert (concaténation de fichiers binaires), le dernier rencontré par la procédure de chargement, sera celui qui sera utilisé, les autres seront ignorés.

Pour lire ou écrire un fichier binaire à travers le FMS, directement à partir d'un programme utilisateur, le programme appelant doit lui même gérer les octets d'informations en en-tête de l'enregistrement ; FLEX ne fournira pas ces informations à l'utilisateur.

7.5 - Description d'un fichier source

Un fichier source (appelé aussi "fichier ASCII", "fichier codé", ou "fichier texte") ne contient que des caractères ASCII imprimables, plus quelques caractères de contrôle spéciaux.

Il n'y a pas d'adresse de chargement pour un fichier source. C'est au programme qui lit le fichier texte, qu'il appartient de mettre les données du fichier à leur place.

Les seuls caractères de contrôle reconnus ou manipulés par FLEX sont :

- \$0D (ASCII : CR ou RETOUR CHARIOT)

Ce caractère est utilisé pour marquer la fin d'une ligne ou d'un enregistrement dans le fichier.

- \$00 (ASCII : NULL)

Ce caractère est ignoré par FLEX. S'il est rencontré dans un fichier, il ne sera pas renvoyé au programme appelant.

- \$18 (ASCII : ANNULER)

Ce caractère est ignoré par FLEX. S'il est rencontré dans un fichier, il ne sera pas renvoyé au programme appelant.

- \$09 (ASCII : HT ou TABULATION HORIZONTALE)

C'est un caractère drapeau qui indique qu'une chaîne d'espace a été retirée du fichier. L'octet suivant ce caractère contient le nombre d'espaces supprimés (2-127).

Le programme appelant ne voit ni le caractère drapeau, ni le compteur de caractères. Le nombre exact d'espaces est retourné au programme utilisateur caractère par caractère à chaque exécution de la fonction "lecture du prochain caractère" (fonction 0).

A l'écriture d'un fichier, les espaces sont automatiquement effacés au fur et à mesure qu'ils sont envoyés au FMS par la fonction "Ecriture du prochain caractère". La compression des données est donc totalement transparente pour le programme appelant.

(Cette description n'est valable que pour des fichiers sources. Si le fichier ouvert est un fichier binaire, le caractère drapeau et le compteur seront transmis identiques, à ce qu'ils sont dans le fichier).

7.6 - Création de commandes utilitaires

Les commandes utilitaires peuvent être réalisées en assembleur. FLEX réserve un espace mémoire dans lequel des utilitaires de taille moyenne peuvent être chargés.

Cette zone mémoire commence en \$ C100 (hex) et peut s'étendre jusqu'en \$ C6FF. Le FCB système localisé en \$ C840 peut être aussi utilisé par les commandes utilitaires créées par le programmeur, comme zone de FCB ou de stockage temporaire. Il ne doit pas y avoir de données résidant dans cette zone de FCB tant qu'il peut y avoir des interférences avec le chargement des programmes utilitaires (FLEX utilise cette zone pour charger les commandes utilitaires en mémoire).

Un exemple sera donné plus loin en démonstration des conventions et des techniques à utiliser pour créer des commandes utilitaires.

Cet exemple dont on trouvera le listing dans les pages suivantes est simplement un utilitaire permettant de lister un fichier texte sur l'écran.

La syntaxe de la commande est :

LIST, [<identification du fichier>]

L'extension spécifiée par défaut est TXT. Le contenu du fichier sera affiché sur le terminal ligne par ligne.

DESCRIPTION DU LISTING SOURCE DE L'UTILITAIRE LIST

1) EQUIVALENCES FLEX

C'est un ensemble de paramètres indiquants à l'assembleur l'emplacement mémoire des différents sous-programmes du FLEX utilisés.

Ces paramètres font partie des adresses données dans ce manuel, dans le chapitre "sous-programmes système appelables par l'Utilisateur".

2) EQUIVALENCES FMS

Ce sont les points d'entrée du FMS, utilisés au cours de l'exécution du programme.

3) EQUIVALENCES SYSTEME

Cette section contient l'adresse de début du bloc de contrôle du fichier (FCB) utilisé.

L'adresse de début de stockage du programme est donnée par l'instruction ORG. La zone mémoire réservée aux commandes utilitaires est utilisée, aussi l'origine du programme est en \$ C100.

4) PROGRAMME

Une des premières conventions à respecter à l'écriture d'une commande utilitaire est de toujours commencer le programme par une instruction BRA suivie d'une ligne de type "VN FCB 1" qui définira le numéro de version de l'utilitaire.

Dans cet exemple, le numéro de version est 1.

Cette convention permet l'utilisation correcte de la commande utilitaire VERSION.

En "LIST 2", le programme fait l'acquisition des paramètres d'identification du fichier et les stocke dans le FCB. Le sous-programme GETFIL est utilisé en plus de l'acquisition et de la sauvegarde des informations, il teste d'éventuelles erreurs. Si tout se passe bien, le bit de retenue (c) dans le registre de condition du CPU est mis à "0", autrement, une ou plusieurs erreurs ont été détectées dans le nom du fichier (C=1) et le contrôle est passé à la ligne d'étiquette "LIST 9". A cet endroit, un message d'erreur est affiché et la main est redonnée au système FLEX.

Si le nom du fichier est correct et que le bit de retenue du registre de condition est à zéro, une extension TXT est donnée par défaut au nom du fichier. Il suffit de mettre le code de l'extension TXT dans l'accumulateur A (TXT=1) et d'appeler le sous-programme SETEXT résidant dans le DØS. Le registre d'index doit pointer sur l'adresse du FCB choisi. Le code "1" est aussi chargé dans l'octet de code fonction du FCB en prévision de l'opération à effectuer sur le fichier (ouverture en lecture). Il n'y a pas d'erreurs générées par un appel à SETEXT.

Maintenant, le fichier peut être ouvert en lecture, le code fonction correspondant ayant déjà été chargé dans le FCB et le registre d'index du CPU pointant toujours sur l'adresse de début du FCB, il suffit d'appeler le FMS.

Si une erreur est détectée, l'instruction "BNE LIST 9" passera le contrôle au module de gestion de l'erreur.

La première chose que celui-ci fait, c'est d'appeler le sous-programme du DØS, RPTERR qui affichera sur le terminal le message correspondant au code de l'erreur détectée. Ensuite tous les fichiers ouverts seront fermés ; ceci est facilement réalisé en appelant le point d'entrée du FMS : "fermeture des fichiers" (FMSCLS).

Finalement, le contrôle est redonné au DØS au point d'entrée à chaud.

Si l'ouverture du fichier s'est déroulée avec succès, le programme continue à la ligne d'étiquette LIST4". A ce moment, il est possible de faire l'acquisition des caractères un par un à l'intérieur du fichier, et de les afficher sur le terminal. Les "Retour Chariot" ne sont pas stockés dans un fichier texte. (Ils marquent la fin d'une ligne, mais la ligne suivante suit immédiatement après), aussi tous les "Retour Chariot" reçus par le système ne sont pas affichés. Pour passer à la ligne à la réception du caractère "Retour Chariot", il suffit de faire appel au sous-programme du DØS - "PCRLF".

Chaque fois qu'un caractère venant du fichier est reçu par le système (appel du FMS en LIST4), un test d'erreur est exécuté. Si une erreur est détectée, le contrôle est passé à ligne d'étiquette "LIST6". Comme le FLEX ne sauvegarde pas un caractère de "Fin de Fichier" dans le fichier, la seule façon de déterminer la fin du fichier est de tester l'erreur "Fin de Fichier" générée par le FMS (code erreur 8). Si le code erreur envoyé n'est pas égal à 8, le contrôle est passé au module de gestion de l'erreur décrit plus haut. S'il est égal à 8, l'édition du fichier est terminée, aussi doit-il être fermé. Le code fonction FMS de fermeture de fichier est 4. Il est chargé dans l'accumulateur A et stocké dans le FCB, un appel au FMS suffira alors pour fermer le fichier. En retour, les erreurs sont testées et s'il n'y en a pas, la main est redonnée au DØS en appelant le point d'entrée à chaud du FLEX.

Cet exemple illustre une des méthodes utilisées pour créer un utilitaire. De nombreuses routines du DØS et du FMS sont appelées. Le concept de base de l'ouverture ou de la fermeture d'un fichier est démontré, ainsi que celui des Entrées/Sorties. La méthode de traitement des erreurs est aussi exposée.

L'étude approfondie de cet exemple permettra à l'utilisateur de comprendre parfaitement les techniques de création des commande utilitaires.

COMMANDE UTILITAIRE = LIST

COPYRIGHT (C) 1978 BY

TECHNICAL SYSTEMS COMMUTANTS INC.

* EQUIVALENCES D/S

CD03	WARMS	EQU	\$CD03	POINT D'ENTREE A CHAUD
CD2D	GETFIL	EQU	\$CD2D	ACQUI. SPECIF. FICHIER
CD18	PUTCHR	EQU	\$CD18	SORTIE DU CARACTERE
CD24	PCRLF	EQU	\$CD24	IMPRESSION SAUT A LA LIGNE
CD33	SETEXT	EQU	\$CD33	INIT. EXTENSION PAR DEFAULT
CD3F	RPTERR	EQU	\$CD3F	AFFICHAGE ERREURS DISQUE

* EQUIVALENCES FMS

D406	FMS	EQU	\$D406
D403	FMSCLS	EQU	\$D403

* EQUIVALENCES SYSTEME

C840	FCB	EQU	\$C840	FCB SYSTEME
------	-----	-----	--------	-------------

* DEBUT DU PROGRAMME

C100		ØRG	\$C100	DEBUT
C100 20 01	LIST	BRA	LIST2	NUMERO DE LA VERSION
C102 01	VN	FCB	1	POINTE SUR L'ADRESSE DU FCB
C103 CE C8 40	LIST2	LDX	#FCR	ACQUIS. SPECIF. FICHIER
C106 BD CD 20		JSR	GETFIL	ERREURS ???
C109 25 34		BCS	LIST9	CODE TXT OU LECTURE
C10B 86 01		LDA A	#1	SAUVEGARDE DANS LE FCB
C10D A7 00		STA A	O,X A	EXTENSION PAR DEFAULT
C10F BD CD 33		JSR	SETEXT	APPEL DU FMS. OUVERTURE DU FICHIER
C112 BD B4 06		JSR	FMS	TEST D'ERREUR
C115 26 28		BNE	LIST9	POINTE SUR L'ADRESSE DU FCB
C117 CE C8 40	LIST4	LDX	#FCB	APPEL DU FMS. ACQUIS. D'UN CARACT.
C11A BD D4 06		JSR	FMS	ERREURS ???
C11D 26 0E		BNE	LIST6	RETOUR CHARIOT ???
C11F 31 0D		CMP A	#SD	
C121 26 05		BNE	LIST5	
C123 BD CD 24		JSR	PCHLF	SORTIE SAUT A LA LIGNE
C126 20 EF		BRA	LIST4	REPETITION
C128 BD CD 18	LIST5	JSR	PUTCHR	SORTIE DU CARACTERE
C12B 20 EA		BRA	LIST4	REPETITION
C12D A6 01	LIST6	LDA A	1,X	ACQUIS. CODE ERREUR
C12F 81 08		CMP A	#8	ERREUR FIN DE FICHIER ???
C131 26 0C		BNE	LIST9	
C133 86 04		LDA A	#4	CODE FERMETURE DU FICHIER
C135 A7 00		STA A	O,X	SAUVEGARDE DANS LE FCB
C137 BD D4 06		JSR	FMS	APPEL DU FMS. FERMETURE
C13A 26 03		BNE	LIST9	ERREURS ???
C13C 7E CD 03		JMP	WARMS	RETOUR SOUS FLEX
C13F BD CD 3F	LIST9	JSR	RPTERR	AFFICHAGE DE L'ERREUR
C142 BD D4 03		JSR	FMSCLS	FERMETURE DE TOUS LES FICHIERS
C145 7E CD 03		JMP	WARMS	RETOUR SOUS FLEX

END LIST

S M T

7.7 - Commande utilitaire : LINK

L'utilitaire LINK fourni sur la disquette FLEX est une commande spéciale. Sa seule fonction est de préciser au "chargeur binaire", qui se trouve sur la piste 00, où se trouve le système FLEX sur le disque.

Le FLEX pouvant résider n'importe où sur le disque, LINK prend son adresse physique sur le disque, et la stocke dans un des pointeurs des secteurs du chargeur.

Quand le programme s'exécute, le chargeur lit simplement cette adresse et charge en mémoire le fichier binaire qui se trouve à cet endroit. Le processus de chargement s'arrête à la réception d'une adresse de transfert. A ce moment, le contrôle est passé au programme venant d'être chargé par une instruction de saut (JMP) à l'adresse de transfert reçue. Si jamais le programme est déplacé sur le disque après avoir été "relié" au chargeur binaire, il est nécessaire de relancer la commande LINK.

LINK peut être utilisée dans plusieurs applications. L'une d'elles est le développement d'un système d'exploitation propre à l'utilisateur. L'utilisateur peut parfaitement utiliser son propre système d'exploitation, le relier au chargeur binaire et l'utiliser à la place du FLEX.

Il peut être aussi intéressant dans des configurations spéciales de pouvoir charger quelques modules spécialisés avant le FLEX. Si cela est le cas, il faut se rappeler que tant que le FLEX n'a pas été chargé en mémoire, il n'y a ni module d'exploitation des disques résidents en MEV, ni système de gestion de fichiers.

7.8 - Modules d'impression

Deux programmes d'impression sont fournis avec FLEX. Ils ne peuvent se concevoir l'un sans l'autre. Le premier est la commande utilitaire "P" et l'autre est le fichier "PRINT.SYS" (initialisation du terminal imprimant et sortie du caractère).

Le listing source de la commande "P" est fourni dans les pages suivantes.

Le fichier PRINT.SYS peut être modifié afin de s'adapter au terminal choisi par l'utilisateur. On trouvera ci-après les impératifs à suivre à la création du fichier PRINT.SYS.

Le listing source de celui fourni avec la disquette FLEX (interface parallèle de type CENTRONICS) se trouve dans le "Manuel d'utilisation FLEX."

CARACTERISTIQUES DU FICHIER "PRINT.SYS"

Le fichier doit fournir au système au moins trois sous-programmes de base.

- un sous-programme d'initialisation de l'imprimante
- un sous-programme de test de l'imprimante
- un sous-programme de sortie du caractère

La commande "P" et le handler physique d'impression utilisent ces trois sous-programmes pour communiquer avec l'imprimante. Les trois modules et leurs caractéristiques sont décrits ici :

PINIT : (\$CCCO-CCD7) : ce module sert à initialiser le port de l'imprimante. Aucun des registres du CPU n'a besoin d'être sauvegardé.

PCHK : (\$CCD8-CCE3) : ce module teste l'état de l'imprimante pour voir si elle est prête à accepter un autre caractère. Un code condition négatif est renvoyée si elle n'est pas libre, positif dans l'autre cas. Les registres A, B et X ne doivent pas être modifiés en sortie.

PØUT : (\$CCE4-CCF7) : ce module sort sur l'imprimante le caractère contenu dans l'accu. A après avoir testé si l'imprimante peut le recevoir (PCHK). Les registres B et X ne doivent pas être modifiés.

MODULE D'IMPRESSION SIMULTANEE

FLEX contient un module d'impression simultanée ("Spooling") Essentiellement, ce module est un système multi-tâche, la fonction d'impression étant une tâche à priorité de bas niveau.

Chaque demande d'accès au disque interrompt l'impression tant que le disque est utilisé. Il faut noter que les vecteurs d'interruption SWI et LRQ sont utilisés par le "spooler". La commande PRINT est utilisée pour mettre en route ce module qui imprimera tour à tour les fichiers contenus dans la queue d'impression.

* COMMANDE UTILITAIRE "P"

* LA COMMANDE "P" INITIALISE UN PORT D'E/S ET MODIFIE LE VECTEUR DE SAUT A OUTCH DANS FLEX.

1	OPT	PAG, NOG
2	TTL	P COMMAND

P COMMAND 26-11-81 ASSEMBLEUR TSC 6800 PAGE 1

```

4
5      * COPYRIGHT 1978 BY TSC
6
7      * EQUATES
8
9 0010      INDEX EQU $0010
10 C840      FCB EQU $C840
11 CD30      LOAD EQU $CD30
12 D406      FMS EQU $D406
13 D403      FMSCLS EQU $D403
14 CD06      RENTER EQU $CD06
15 0004      NFER EQU $4
16 CC09      PAUSE EQU $CC09
17 CD1E      PSTANG EQU $CD1E
18 CD3F      RPTERR EQU $CD3F
19 CD03      WARMS EQU $CD03
20 CC11      LSTTRM EQU $CC11
21 CC02      EOL EQU $CC02
22 CCC0      PINIT EQU $CCC0
23 CCE4      POUT EQU $CCE4
24 CD0F      OUTCH EQU $CD0F
25 CCFC      PR1 EQU $CCFC
26
27 C100      ORG $C100
28
29 C100 20 01      P      BRA P1
30
31 C102 01      VN      FCB 1
32
33 C103 06 CC FC P1      LDA A PR1
34 C106 27 09      BEQ P12
35 C108 CE C8 40      LDX $FCB
36 C10B C6 1B      LDA B $27
37 C10D E7 01      STA B 1,X
38 C10F 20 43      BRA P3
39 C111 06 CC 11 P12      LDA A LSTTRM
40 C114 81 0D      CMP A $D
41 C116 27 45      BEQ P8
42 C118 B1 CC 02      CMP A EOL
43 C11B 27 40      BEQ P8
44 C11D 7F CC 09      CLR PAUSE
45 C120 B6 CC E4      LDA A POUT
46 C123 81 39      CMP A $39
47 C125 26 13      BNE P15
48 C127 CE C8 40      LDX $FCB
49 C12A 86 01      LDA A $1
50 C12C A7 00      STA A 0,X
51 C12E BD D4 06      JSR FMS
52 C131 26 13      BNE P2
53 C133 86 FF      LDA A $FF
54 C135 A7 3B      STA A $3B,X
55 C137 BD CD 30      JSR LOAD
56 C13A BD CC C0 P15      JSR PINIT
57 C13D CE CC E4      LDX $POUT
58 C140 FF CD 10      STX OUTCH+1

```

P COMMAND

25-11-81 ASSEMBLEUR TSC 68000 PAGE 2

```

59 C143 7E CD 06      JMP    RENTER
60 C146 A6 01      P2    LDA A 1,X
61 C148 81 04      CMP A ENFER
62 C14A 26 08      BNE    P3
63 C14C CE C1 62      LDX    ENDPST
64 C14F BD CD 1E P25 JSR    PSTANG
65 C152 20 03      BRA    P4
66 C154 BD CD 3F P3 JSR    RPTERR
67 C157 BD D4 03 P4 JSR    FMSCLS
68 C15A 7E CD 03      JMP    WARMS
69 C15D CE C1 79 P8 LDX    ERSTR
70 C160 20 ED      BRA    P25
71
72 C162 22      NOPST FCC    'PRINT.SYS' non trouvé'
73 C178 04      FCB    4
74 C179 22      ERSTR FCC    /"P" doit etre suivie d'une commande/
75 C19C 04      FCB    4
76
77 C843      ORG    #C843
78
79 C843 FF      FCB    $FF
80 C844 50      FCC    'PRINT'
81 C849 00      FCB    0,0,0
82 C84C 53      FCC    'SYS'
83
84      END    P

```

AUCUNE ERREUR DETECTEE

P COMMAND

25-11-81 ASSEMBLEUR TSC 68000 PAGE 3

TABLE SYMBOLES

EDL	CC02	ERSTR	C179	FCB	C840	FMS	D406	FMSCLS	D403
INDEX	0010	LOAD	CD30	LSTRM	CC11	NFER	0004	NOPST	C162
OUTCH	CD0F	P	C100	P1	C103	P12	C111	P15	C13A
P2	C146	P25	C14F	P3	C154	P4	C157	P8	C15D
PAUSE	CC09	PINIT	CCC0	POUT	CCE4	PR1	CCFC	PSTRNG	CD1E
RENTERR	CD06	RPTERR	CD3F	VN	C102	WARMS	CD03		

* COPYRIGHT (C) 1978 BY

* TECHNICAL SYSTEM CONSULTANTS, INC.

7.9 - Les interruptions dans FLEX

FLEX fait une utilisation intensive des interruptions pendant l'impression simultanée.

Tous les utilitaires du FLEX, l'Editeur, l'Assembleur, le Processeur de texte et le Basic sont interruptibles.

Les programmes utilisateurs peuvent eux-mêmes être interrompus. Mais à aucun moment, les vecteurs d'IRQ ou SWI ne doivent être modifiés dans un utilitaire utilisant les modules d'impression.

CARTES MEMOIRE OU SYSTEME

0000 B FFF	RAM utilisateur
C000 C07F	Pile système
C080 C0FF	Buffer d'entrée
C100 C6FF	Zone des commandes utilitaires
C700 C83F	Superviseur et modules d'impression
C840 C97F	FCB du système
C980 CBFF	Zone des fichiers systèmes
CC00 D3FF	DØS
D400 DE7F	FMS
DE80 DFFF	Modules d'exploitation du disque